INTRODUCCIÓN

En este trabajo pretendemos resaltar la función de Tesorería como elemento indispensable dentro de una organización.

El SISTEMA DE GESTIÓN DE TESORERÍA provee de todos los servicios de cobranza y pagos a los usuarios a través de un sitio Web.

Una buena tesorería no es nunca fruto de la improvisación del momento, se basa en:

- Que la empresa obtenga beneficios.
- Que la gestión financiera y general de la empresa se haya planteado con toda seriedad

La función más importante de Tesorería es la supervisión de cobros y pagos a terceros.

Introducción a los documentos soporte

Cheque: Instrumento privado que constituye una orden de pago sobre un banco en la cual el girador mantiene en su cuenta corriente saldo a su favor, es un mandato de pago que sirve al girador para pagar a un tercero o a si mismo, con los fondos disponibles que tenga en su cuenta corriente.

Factura Conformada: Título de valor aceptado por el cliente, aceptando las condiciones de crédito del proveedor.

Se origina en la compra venta de mercaderías, así como en otras modalidades contractuales de transferencia de la propiedad de bienes susceptibles de ser afectados en prenda. En las que se acuerde el pago diferido del precio.

Nota de Venta: Es un documento que se origina en base a un pedido verbal o por escrito hecho por el comprador, mediante el cual se obliga al vendedor a entregar las mercaderías y al comprador a recibirlas.

Nota de Débito: Este documento se diligencia cuando es necesario informar

al cliente acerca del valor de su deuda o de algún tipo de incremento por diferentes conceptos.

Nota de Crédito: Este documento se diligencia cuando es necesario informar a un cliente que su deuda disminuye por algún concepto.

Comprobante de Ingreso: Tiene como objeto fundamental servir como comprobante de todos los ingresos monetarios y de constancia del pago efectuado por un cliente.

Orden de Pago: Es un documento no negociable de uso interno en el que se indica la autorización que se requiere para efectuar pagos. Se detallan los datos de las facturas de un proveedor y el cheque emitido con su respectivo valor y fecha.

Comprobante de Egreso: Este documento se diligencia cuando, por diferentes conceptos, la empresa realiza erogaciones de dinero. Tiene como objetivo fundamental servir de constancia de los valores entregados por la empresa.

CAPÍTULO 1

1 SISTEMA DE GESTIÓN DE TESORERÍA

1.1 Problemática

El control de gestión de una Empresa va tomando mayor importancia con el transcurso del tiempo a medida que las actividades han ido creciendo pues no solamente se han multiplicado en número sino también en diversidad y complejidad, su ejecución compromete la utilización de cuantiosas sumas de recursos, que pese a lo significativo de la misma resulta escasas al confrontarlas con la multiplicidad de

necesidades de la sociedad.

Para hacer frente a estas necesidades surgieron los ERP (Enterprise Resource Planning) son sistemas transaccionales, es decir están diseñados para trabajar con procesos de la empresa, soportarlos, procesar los datos y obtener de ellos información especifica.

Así, puede haber un seguimiento y control de los procesos de la Empresa, como son: Finanzas y Contabilidad, Ventas, Mercadotecnia, Compras, Recursos Humanos, Operaciones y Abastecimientos.

La función principal es organizar y estandarizar procesos y datos internos de la empresa trasformándolos en información útil para ser analizados para la toma de decisiones.

Es importante recordar que finalmente, aunque estos sistemas apoyan en la toma de decisiones, no quiere decir que ellos lo hagan, sino que los administradores (humanos), tienen el poder final para tomar las decisiones estratégicas y adecuadas en la Empresa.

La utilización de códigos libres (OPEN SOURCE) minimiza el costo de

desarrollo de esta tecnología.

La sección de Tesorería, que se encuentra dentro del departamento Financiero, es un área que tiene un fuerte componente administrativo y operacional que actúa como un lastre para el desempeño de otras funciones mas llegadas a la gestión y a la creación de valor.

1.2 Solución

La necesidad de la empresa comercial de tener un sistema integrado bajo Web, que facilite sus actividades empresariales, al menor costo posible, y con tecnología de punta, ha hecho que la empresa decida implantar un sistema ERP bajo Web usando tecnología OPEN SOURCE.

En los actuales momentos el departamento de Tesorería; que se encarga de recibir, controlar y custodiar ordenadamente los valores así como sus reportes, no cuenta con sistema automatizado.

Es por este motivo que nuestro proyecto de tesis dirigida implementará el módulo “Sistema de Gestión de Tesorería”, el que permitirá receptar cobros, realizar pagos, emisión de comprobantes, informes o

resúmenes de Tesorería.

La base para el desarrollo de los diferentes tópicos de nuestra tesis dirigida serán los requerimientos del departamento de Tesorería de la Empresa Comercial.

1.3 Visión

Trabajar con procesos de la empresa, soportarlos, procesar los datos y obtener de ellos información específica.

1.4 Misión

Para poder obtener todo esto será necesario que todos los departamentos de sistemas estén conectados o que exista un comunicación directa entre departamentos, para así se ahorre tiempo, exista coordinación, no exista gastos innecesarios, para así la empresa sea la mas beneficiada en este tema. La función principal es organizar y estandarizar procesos y datos internos de la empresa trasformándolos en información útil para ser analizados para la toma de decisiones.

El módulo “Sistema de Gestión de Tesorería”, el que permitirá receptar cobros, emisión de comprobantes, informes o resúmenes de Tesorería

y emitir pagos.

1.5 Objetivos generales

A continuación se detallan los objetivos generales, del modulo SISTEMA DE GESTION DE TESORERIA, para su realización y desarrollo:

- ❖ Investigar Tecnología OPEN SOURCE, tales como J2EE, BASES de DATOS como PostgreSQL y MySQL, Herramientas para la creación de reportes tal como iREPORT.

- ❖ Analizar, desarrollar e implementar el modulo de Tesorería (Sistema de Gestión de Tesorería) bajo Web, usando Herramientas OPEN SOURCE.

- ❖ Ofrecer las seguridades necesarias, para que el acceso a la información del sitio Web solo sea posible para las personas que estén vinculadas directamente con la empresa.

1.6 Objetivos específicos

Para el desarrollo del SISTEMA DE GESTIÓN DE TESORERÍA, hemos

considerado los siguientes objetivos específicos:

- ❖ Permitir el uso de la aplicación SISTEMA DE GESTIÓN DE TESORERÍA solo a los usuarios autorizados.

- ❖ Asignar a los usuarios perfiles de acceso a la información del SISTEMA DE GESTIÓN DE TESORERÍA.

- ❖ Desarrollar e implementar los requerimientos mas importantes del sitio Web, que permitirá obtener y presentar una información adecuada.

- ❖ Permitir la emisión de comprobantes, informes o resúmenes de Tesorería.

1.7 Beneficios

Los beneficios que aportará el SISTEMA DE GESTIÓN DE TESORERÍA son los siguientes:

- ❖ El proceso del departamento de Tesorería llegaría a realizar sus

actividades de manera rápida y fiable.

- El módulo permitirá la emisión de comprobantes, informes o resúmenes de Tesorería.

- Agilitar el trabajo del departamento de Tesorería, reduciéndose la aglomeración de usuarios.

- La función principal es organizar y estandarizar procesos y datos internos de la empresa trasformándolos en información útil para ser analizados para la toma de decisiones.

1.8 Alcance

Para que este proyecto cumpla con los objetivos anteriormente planteados hemos definido las siguientes actividades o metas:

- Sub modulo de Pagaduría.
  - Emisión de Ordenes de Pagos.
  - Reporte para el pago de proveedores.
  - Pago en varios medios (Cheques, transferencias, tarjeta, etc.).

- Permite la cancelación de varias facturas que pertenecen a un proveedor con un pago.
- Emisión de cheques en formularios preimpresos.
- Administración de Cheques rechazados por varios motivos.
- Anulación de cheques.
- Reporte de Transferencias.
- Reportes de Cheques.

❖ Sub modulo de Recaudaciones

- Reporte de cobros de cartera para realizar depósito de lo recaudado.
- Reporte de comprobantes de ingresos.
- Cobros en varios medios (cheques, tarjeta, etc.).
- Administración de cheques post fechados.

1.9 Cronograma

A continuación se detallan las tareas a realizar, de manera resumida y detallada, para el desarrollo del SISTEMA DE GESTIÓN DE TESARERÍA.

1.9.1 Cronograma resumido

No	Tarea	Inicio	Fin
1	Levantamiento de información	23-Ene-06	10-Feb-06
2	Auto capacitación	13-Feb-06	04-Mar-06
3	Asesoramiento profesional	06-Mar-06	17-Mar-06
4	Análisis	20-Mar-06	21-Abr-06
5	Documentación del análisis	24-Abr-06	05-May-06
6	Diseño	08-May-06	16-Jun-06
7	Documentación del diseño	19-Jun-06	30-Jun-06
8	Implementación	03-Jul-06	25-Ago-06
9	Documentación de la implementación	28-Ago-06	08-Sep-06
10	Pruebas y depuración	11-Sep-06	29-Sep-06
11	Documentacion pruebas, manual técnico y de usuario	02-Oct-06	12-Oct-06

Tabla 1. Cronograma resumido

1.9.2 Cronograma detallado

El cronograma detallado se describe en el anexo A

1.10 Recursos

Los recursos humanos, de hardware y de software para el desarrollo del SISTEMA DE GESTIÓN DE TESORERÍA se detallan a continuación:

1.10.1 Recursos humanos

Esta conformado por la participación de Robert Roca, líder de Proyecto, Luisa Castro, DBA, y Fabricio Villalta, analista – programador.

Cada uno prestará sus servicios durante 6 meses.

Cantidad	Detalle	V. Unit.	V. total
1	Líder de Proyecto	400,00	2400,00
1	DBA	300,00	1800,00
1	Analista – programador	300,00	1800,00
	Total		6000,00

Tabla 2. Recursos humanos

1.10.2 Recursos de hardware

Para el desarrollo del sitio Web se requirió del alquiler de 2

máquinas para los 6 meses de duración de nuestro proyecto, además de adquirir una máquina que cumplirá con las funciones de Servidor de aplicaciones y de Servidor de Base de Datos.

Cantidad	Detalle	Valor
2	Pentium IV, 256 RAM, HD 80 Gb	900,00
1	Pentium IV, 512 RAM, 120 Gb	800,00
	Total	1700,00

Tabla 3. Recursos de hardware

1.10.3 Recursos de software

Se requiere herramientas de programación y de base de datos para el desarrollo e implantación del sistema.

Cantidad	Detalle	V. total
3	Fedora Core 4	0,00
2	Eclipse	0,00
2	Ireport	0,00
2	J2EE	0,00

Cantidad	Detalle	V. total
1	Apache TomCat	0,00
1	Objecteering/UML Modeler	0,00
1	Base de Datos PostgreSql	0,00
	Total	00,00

Tabla 4. Recursos de software

1.10.4 Total de presupuesto para la adquisición de los recursos

Detalle	Valor
Recursos humanos	6000,00
Recursos de hardware	1700,00
Recursos de software	0,00
Total	7700,00

Tabla 5. Presupuesto de recursos

1.11 Metodología

La metodología se basa en el desarrollo orientado objetos utilizando el Lenguaje Modelado Unificado (UML) que considera las siguientes

etapas:

- Análisis.
- Diseño.
- Implementación.
- Pruebas y depuración.

1.11.1 Metodología del análisis

1.11.1.1 Especificaciones de requisitos del sistema (SRS)

Una descripción textual del alcance y la misión general del sistema.

1.11.1.2 Casos de uso

Una descripción gráfica/textual de cómo se comportará el sistema desde la perspectiva del usuario. Los usuarios pueden ser humanos u otros sistemas.

1.11.1.3 Diagrama de clases

Una representación visual de los objetos que se utilizarán

para construir el sistema.

1.11.2 Metodología del Diseño

1.11.2.1 Escenarios

Descripción textual de los procesos internos necesarios para implementar la funcionalidad documentada en un caso de uso.

1.11.2.2 Diagrama de secuencia

Un modelo de la secuencia de interacciones que ocurren entre los objetos durante la ejecución del programa. Se supone especial énfasis en el orden en el que ocurre las interacciones y cómo evolucionan en el tiempo.

1.11.2.3 Diagrama de actividad

Es una representación visual del flujo de ejecución de un programa u operación**.**

1.11.2.4 Prototipo de la interfaz

Boceto de las diferentes pantallas que compondrán la

interfaz.

1.11.3 Metodología de la implementación

En la etapa de implementación utilizaremos el patrón de arquitectura MVC (Model-View-Controller).

1.11.3.1 Modelo

Se enfoca en la creación de clases JavaBeans que soporten todos los requerimientos de funcionalidad.

1.11.3.1.1 Componentes de la lógica de negocio

Son los encargados de implementar la lógica de negocio para mantener las diversas entidades del sistema, como pueden ser las atracciones, actividades, etc. También se pueden denominar Objetos de negocio.

1.11.3.1.2 Componentes de estado del sistema

Ofrecen los objetos del estado del sistema almacenado en el modelo relacional base.

También se denominan Entidades.

1.11.3.1.3 Componentes lógicos de acceso a datos

Implementan la lógica de acceso a datos para transformar los registros de la base de datos relacional en componentes de estado del sistema y almacenar el estado de los componentes de estado del sistema de nuevo en la base de datos. Este tipo de componentes se denomina Objeto de acceso a datos.

1.11.3.2 Vista

Interfaz de usuario generalmente construida usando tecnología JavaServer Page (JSP). Las páginas JSP pueden contener texto HTML estático (o XML) llamado "plantilla de texto", además de la habilidad de insertar contenido dinámico.

1.11.3.3 Controlador

Está enfocado es las solicitudes recibidas desde el cliente,

decidiendo que función de la lógica de negocio se va a realizar.

1.11.4 Metodología de pruebas

La metodología de Pruebas Orientada a Objetos para el Ciclo de Vida Completo(en ingles "Full Life-Cycle Object-Oriented Testing", FLOOT) es una colección de técnicas para verificar y validar software orientado a objetos disponibles en todos los aspectos del desarrollo de software. Se puede realizar pruebas en todos los aspectos del desarrollo de software no solamente durante la codificación.

CAPÍTULO 2

2 ANÁLISIS

2.1 Especificaciones de Requisitos de Software (SRS)

2.1.1 Usuarios del Sistema

Hemos identificado los siguientes usuarios del sistema:

Tesorero: Persona encargada de recaudar y emplear los caudales en una empresa.

Pagador: Persona que paga las obligaciones de una empresa.

Recaudador: Encargado de realizar la cobranza de los valores de una empresa.

2.1.2 Requisitos del sistema

- Los usuarios tienen que autenticarse utilizando un nombre de usuario y contraseña.

- Las formas de pago son: tarjeta de crédito, transferencias bancarias, efectivo y comprobante de retención.

- Los valores cobrados indebidamente serán devueltos a los contribuyentes o compensados con otras obligaciones.

- Los ingresos recaudados serán revisados, depositados y registrados en la cuenta corriente a nombre de la entidad en un banco durante el día en curso de la recaudación o máximo el día hábil siguiente.

- Está prohibido cambiar cheques, efectuar pagos o préstamos

con los dineros producto de la recaudación.

- ❖ Sobre los valores que se recauden se entregará un recibo prenumerado, fechado, legalizado y con la explicación del concepto y el valor cobrado en letras y números, con sello de cancelado, documento que respaldará la transacción realizada. El comprobante original será entregado a la persona que realice el pago.

- ❖ Diariamente se elaborará el reporte de recaudación.

- ❖ Se efectuará una verificación diaria, con la finalidad de comprobar que los depósitos efectuados sean iguales a los valores recaudados.

- ❖ Efectuada la verificación diaria, si el resultado da una diferencia en más, el valor quedará a favor de la entidad; de producirse una diferencia en menos, si después de un análisis se comprueba que efectivamente es producto de un error la asume la entidad, caso contrario lo asume el recaudador.

- Los valores en efectivo, incluyendo los que se encuentran en poder de los recaudadores de la entidad, estarán sujetos a verificaciones mediante arqueos periódicos.

- Todo desembolso se efectuará mediante cheque, a la orden del beneficiario y por el valor exacto de la obligación que conste en los documentos comprobatorios.

- Las personas autorizadas para girar cheque no tendrán funciones de recaudación de recursos financieros, de recepción de recursos materiales, de registro contable ni de autorización de gastos.

- Las remuneraciones serán pagadas a los servidores con cheques individuales o mediante depósitos directos en sus cuentas corrientes o de ahorros.

- Se cancelan deudas por medio de orden de pago bajo autorización.

- Administración de cheques

2.2 Casos de Uso

2.2.1 Diagrama de Casos de Uso

El diagrama de casos de uso se describe en el anexo B.

2.2.2 Descripción de Casos de Uso

2.2.2.1 Autenticación

Descripción: Se presenta a los usuarios una pantalla de autenticación. Ellos introducen su nombre de usuario y contraseña. A continuación, pueden pulsar sobre conectar o cancelar.

Condiciones previas: Ninguna.

Condiciones posteriores: Acceso al sistema.

2.2.2.2 Consultar estado de cuenta

Condiciones previas: Usuario ha sido autenticado y se ha asignado el estatus (función o rol) de usuario.

Descripción: El recaudador consulta el estado de cuenta. El recaudador introduce la información de búsqueda de la deuda. Tras enviar la petición de búsqueda, se presenta al recaudador el estado de cuenta que coincide con el criterio de búsqueda.

Condiciones posteriores: Recaudador tiene la posibilidad de imprimir su estado de cuenta.

2.2.2.3 Apertura de caja

Condiciones previas: Usuario ha sido autenticado y se ha asignado el estatus (función o rol) de usuario.

Descripción: Definir cual de los turnos de caja activar para el registro de los movimientos.

Condiciones posteriores: El usuario, según el estatus, realizará las tareas de recaudación o pagaduría.

2.2.2.4 Recaudar valores

Condiciones previas: Se ha iniciado la apertura de caja

para la tarea de recaudación.

Descripción: Recaudador procede al cobro de la deuda al cliente haciendo uso de las diferentes formas de pago.

Condiciones posteriores: Se emite al cliente un comprobante de pago que refleja la transacción, además se registra la recaudación a través de un comprobante de ingreso para control interno.

2.2.2.5 Consultar pagos

Condiciones previas: Usuario ha sido autenticado y se ha asignado el estatus (función o rol) de usuario.

Descripción: El Tesorero consulta los pagos. El usuario introduce la información de búsqueda los pagos. Tras enviar la petición de búsqueda se presenta los pagos que coincide con el criterio de búsqueda.

Condiciones posteriores: El Tesorero tiene la posibilidad de imprimir la consulta o generar el pago al proveedor.

2.2.2.6 Generación de pagos

Condiciones previas: El Tesorero ha realizado una consulta de los valores adeudados a un proveedor.

Descripción: El Tesorero define el monto del abono, y la forma de pago mediante la emisión de la orden de pago.

Condiciones posteriores: Emisión de cheque con la información indicada en la orden de pago.

2.2.2.7 Girar cheque

Condiciones previas: Generación y aprobación de orden de pago por el Tesorero.

Descripción: Tesorero emite el cheque por el concepto y valor exacto estipulado en la orden de pago.

Condiciones posteriores: Puede efectuarse el pago a través del pagador.

2.2.2.8 Entrega de valores

Condiciones previas: Se realizó la apertura de caja asignando al usuario la función de pagador.

Descripción: El pagador hace entrega del valor a cancelar definiendo la forma de pago, por lo general a través de cheques. Entregar, según el caso, el comprobante de retención.

Condiciones posteriores: Se realiza un comprobante de egreso para control interno.

2.2.2.9 Anular cheque

Condiciones previas: Giro de cheque.

Descripción: El cheque retorna al departamento de tesorería para su anulación realizada por el tesorero.

Condiciones posteriores: Ninguna.

2.2.2.10 Arqueo de caja

Condición previa: Realización de pagos y recaudación de los valores.

Descripción: Recaudador o pagador realiza la determinación de existencia física y comprobación de igualdad con los saldos contables.

Condiciones posteriores: Reportes de existencia contable.

2.2.2.11 Cierre de caja

Condición previa: Realización de pagos y recaudación de los valores.

Descripción: Recaudador o pagador realiza la determinación de existencia física y comprobación de su igualdad con los saldos contables al final del turno.

Condiciones posteriores: Cierre del turno actual, no se permite registro de movimientos.

2.2.2.12 Depositar valores

Condiciones previas: Recaudación de los valores.

Descripción: Los ingresos recaudados en efectivo, cheques, u otras formas de valor serán revisados, depositados y registrados, por el tesorero, en la cuenta corriente abierta a nombre de la entidad en un banco oficial durante el curso del día de recaudación o máximo el día hábil siguiente.

Condiciones posteriores: se recibe un comprobante de depósito de dicho banco.

2.3 Diagrama de Clases

El diagrama de casos de uso se describe en el anexo C.

CAPÍTULO 3

3 DISEÑO

3.1 Escenarios

3.1.1 Autenticación: Todos los usuarios

1. Se ingresa el usuario.
2. Se ingresa la contraseña
3. El usuario envía una solicitud de autenticación al sistema.
4. Se valida la existencia del usuario

5. Se accede al sistema

3.1.2 Consultar estado de cuenta: Recaudador

1. Se pide identificación del cliente
2. Se valida la identificación del cliente
3. Se muestra de forma detallada el estado de cuenta.

3.1.3 Apertura de caja: Todos los usuarios

1. Especificar la caja y el turno a activar.
2. Validar que la caja y el turno no se encuentren activos.
3. Indicar el monto inicial de la caja.
4. Asignar cajero.
5. Afirmación de la apertura.
6. Confirmación de la apertura.
7. Aceptación de la apertura.

3.1.4 Recaudar valores: Recaudador

1. Ingresar la identificación del cliente.
2. Seleccionar los ítems a cancelar.
3. Seleccionar las formas de pago.
4. Ingresar los valores de cada forma de pago.

5. Afirmación de pago.
6. Confirmación de pago.
7. Aceptación del pago.
8. Se actualiza el estado de cuenta.
9. Se emite comprobante de ingreso.

3.1.5 Consultar pagos: Tesorero

1. Se pide identificación del proveedor.
2. Se valida la identificación del proveedor.
3. Se muestra en forma detallada los valores adeudados.

3.1.6 Generación de pagos: Tesorero

1. Ingresar la identificación del proveedor.
2. Seleccionar los ítems a cancelar.
3. Seleccionar las formas de pago.
4. Ingresar los valores de cada forma de pago.
5. Afirmación de la orden de pago.
6. Confirmación de la orden de pago.
7. Aceptación de la orden del pago.
8. Se actualiza estado de cuenta.
9. Se emite orden de pago aprobada.

3.1.7 Girar cheque: Tesorero

1. Consultar las ordenes de pago aprobadas.
2. Seleccionar orden de pago a cancelar.
3. Verificar datos del proveedor.
4. Verificar datos de los documentos a cancelar.
5. Realizar la emisión del cheque.
6. Aceptación de la emisión del cheque.
7. Confirmación de la emisión del cheque.

3.1.8 Entrega de valores: Pagador

1. Ingresar la identificación del proveedor.
2. Seleccionar la orden de pago.
3. Ingresar beneficiario.
4. Afirmación de pago.
5. Confirmación de pago.
6. Aceptación del pago.
7. Se emite comprobante de egreso.

3.1.9 Anular cheque: Tesorero

1. Consultar cheques emitidos y no entregados al proveedor.
2. Seleccionar cheque a ser anulado.

3. Realizar la anulación del cheque.
4. Aceptación de la anulación del cheque.
5. Confirmar la anulación del cheque
6. Cambiar estado de orden de pago

3.1.10 Arqueo de caja: Todos los usuarios

1. Especificar la caja a consultar.
2. Validar si la caja esta activa.
3. Confirmación de que la caja está activa.
4. Realizar conteo de los valores recaudados mediante los comprobantes de ingreso.
5. Realizar conteo de valores pagados mediante los comprobantes de egreso.
6. Emisión de reporte de existencias contables.

3.1.11 Cierre de caja: Todos los usuarios

1. Especificar la caja a cerrar.
2. Validar si la caja esta activa.
3. Confirmación de que la caja está activa.
4. Realizar conteo de los valores recaudados mediante los comprobantes de ingreso.

5. Realizar conteo de valores pagados mediante los comprobantes de egreso.
6. Emisión de reporte de existencias contables.
7. Cambiar estado de caja a inactiva o cerrada.

3.1.12 Depositar valores: Tesorero

1. Seleccionar la entidad bancaria y la cuenta contable.
2. Clasificar los valores recaudados de acuerdo a su forma de pago.
3. Realizar el conteo de los valores recaudados.
4. Autorización de depósito bancario.
5. Emitir informe de los valores recaudados.

3.2 Diagramas de secuencia

3.2.1 Autenticación: Todos los usuarios

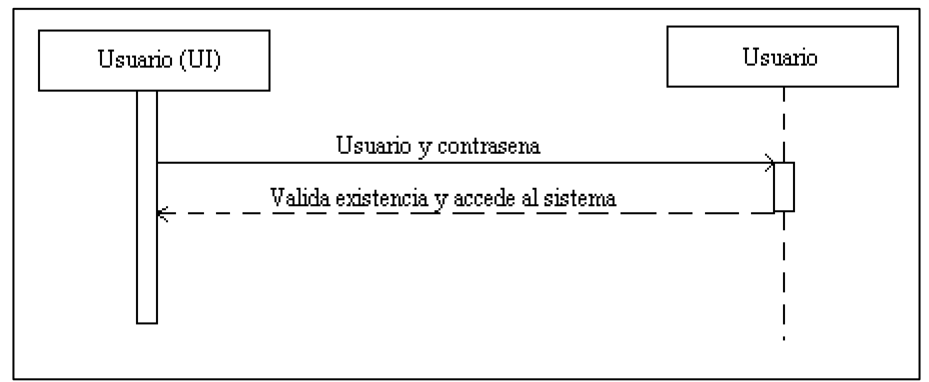


Figura 3.2.1. Diagrama de secuencia de autenticación

3.2.2 Consultar estado de cuenta: Recaudador

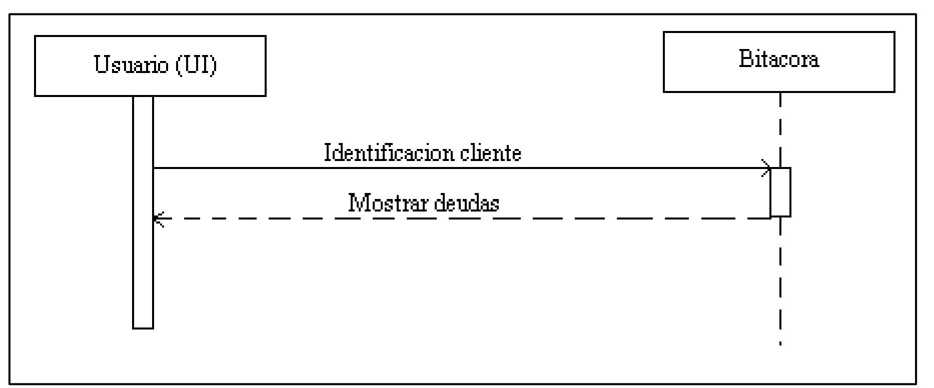


Figura 3.2.2. Diagrama de secuencia de consulta estado de cuenta

3.2.3 Apertura de caja: Todos los usuarios

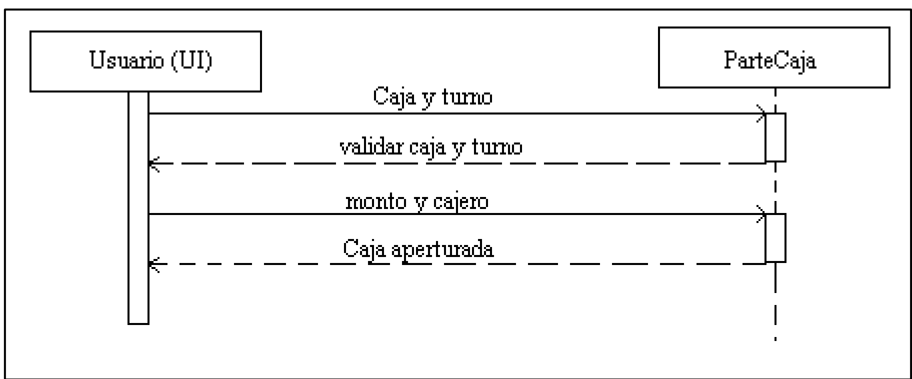


Figura 3.2.3. Diagrama de secuencia de apertura de Caja

3.2.4 Recaudar valores: Recaudador

Figura 3.2.4 Diagrama de secuencia de recaudación valores

3.2.5 Consultar pagos: Tesorero

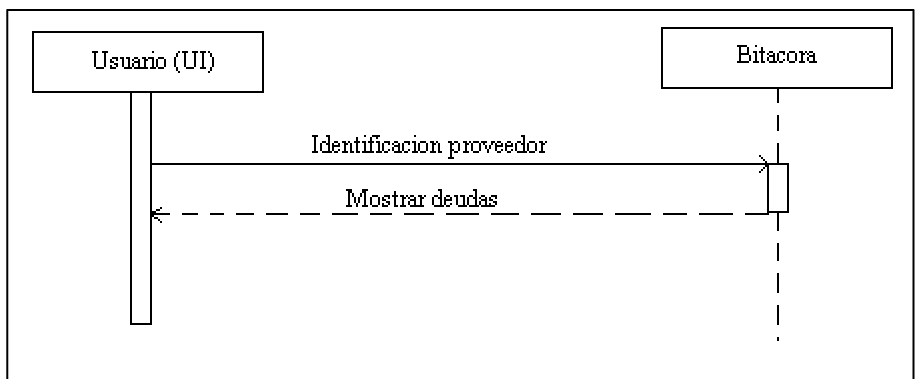


Figura 3.2.5 Diagrama de secuencia de consulta de pagos

3.2.6 Generación de pagos: Tesorero

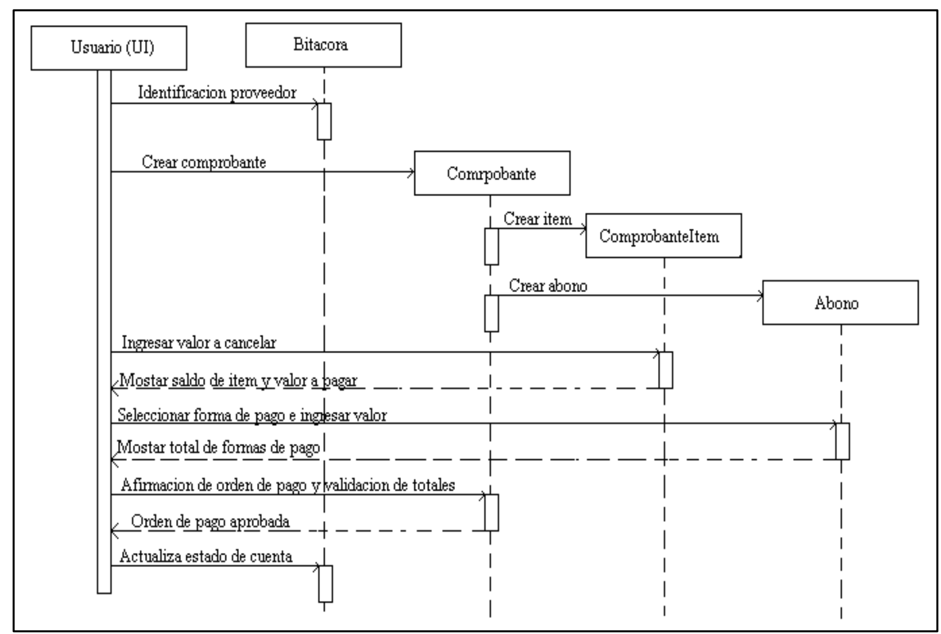


Figura 3.2.6 Diagrama de secuencia de generación de pagos

3.2.7 Girar cheque: Tesorero

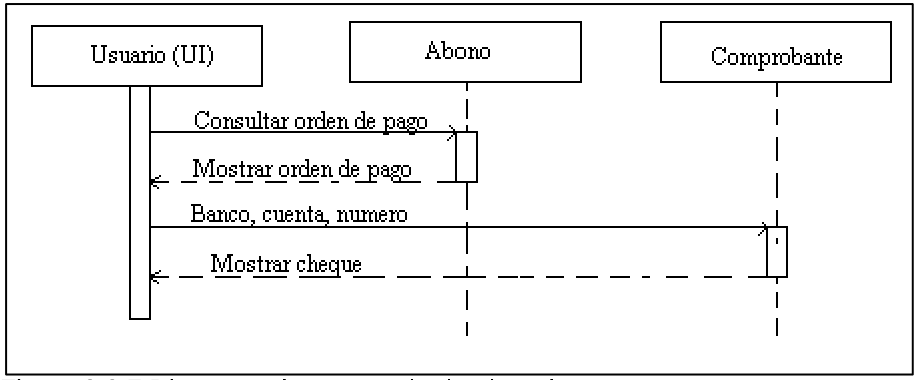


Figura 3.2.7 Diagrama de secuencia de girar cheque

3.2.8 Entrega de valores: Pagador

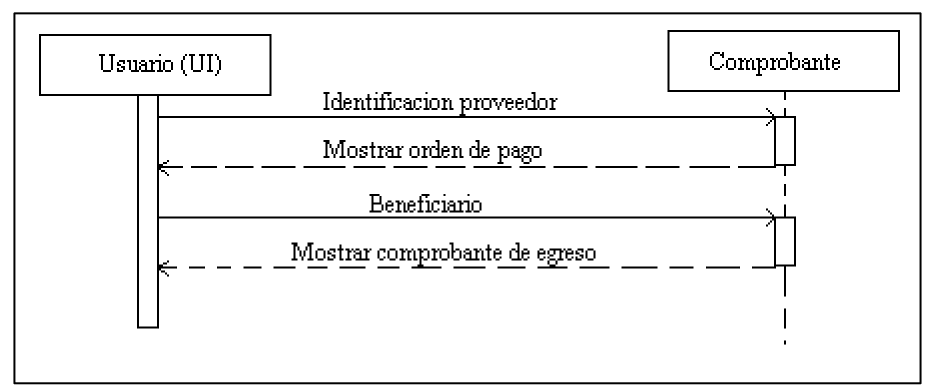


Figura 3.2.9 Diagrama de secuencia de entrega de valores

3.2.9 Anular cheque: Tesorero

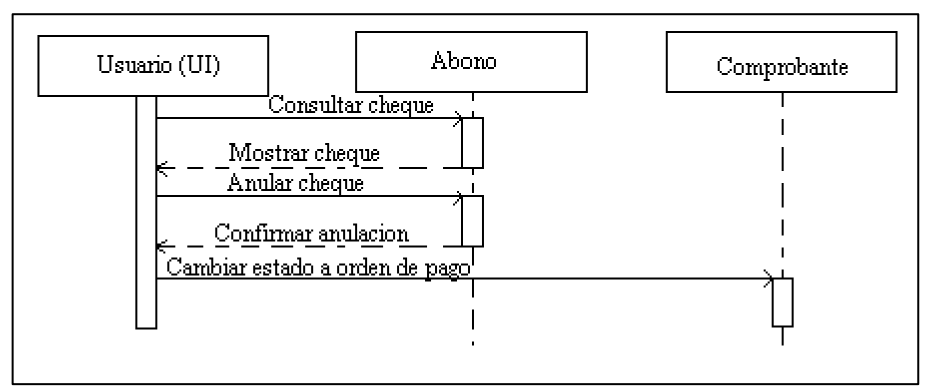


Figura 3.2.9 Diagrama de secuencia de anular cheque

3.2.10 Arqueo de caja: Todos los usuarios

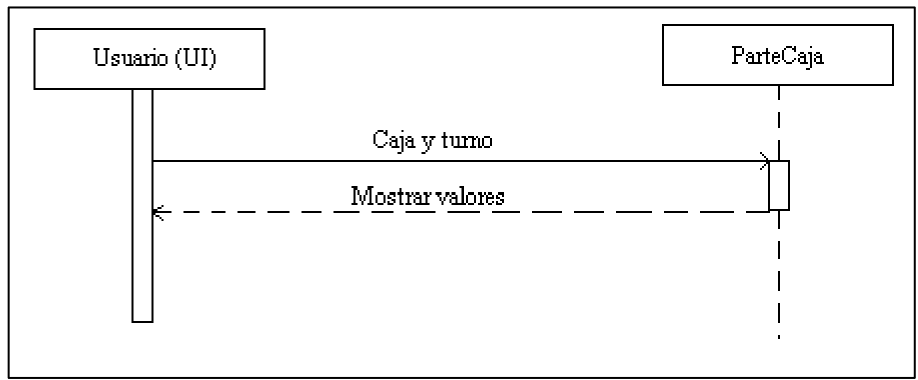


Figura 3.2.10 Diagrama de secuencia de arqueo de caja

3.2.11 Cierre de caja: Todos los usuarios

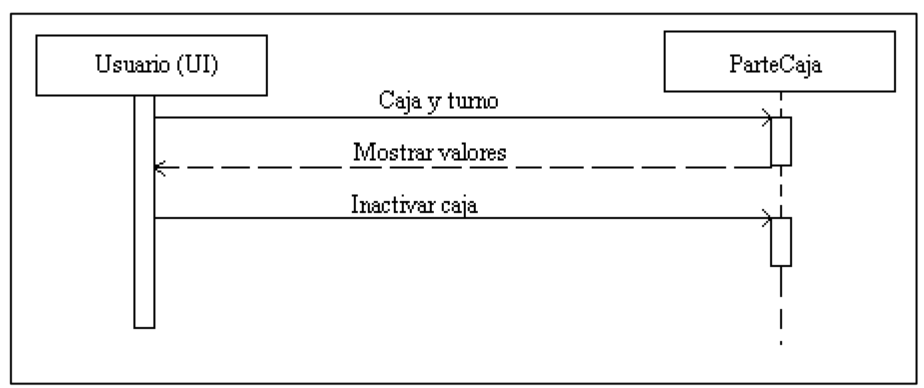


Figura 3.2.11 Diagrama de secuencia de cierre de caja

3.2.12 Depositar valores: Tesorero

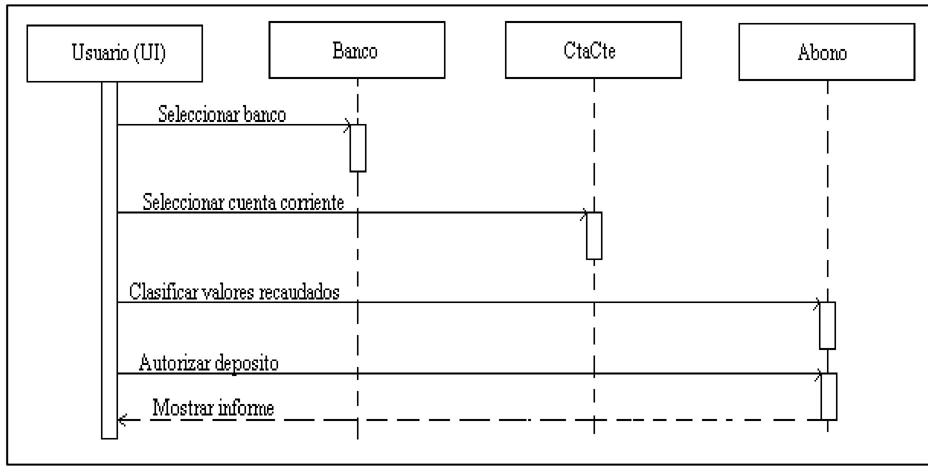


Figura 3.2.12 Diagrama de secuencia de depósito de valores

3.3 Diagramas de actividad

3.3.1 Autenticación: Todos los usuarios

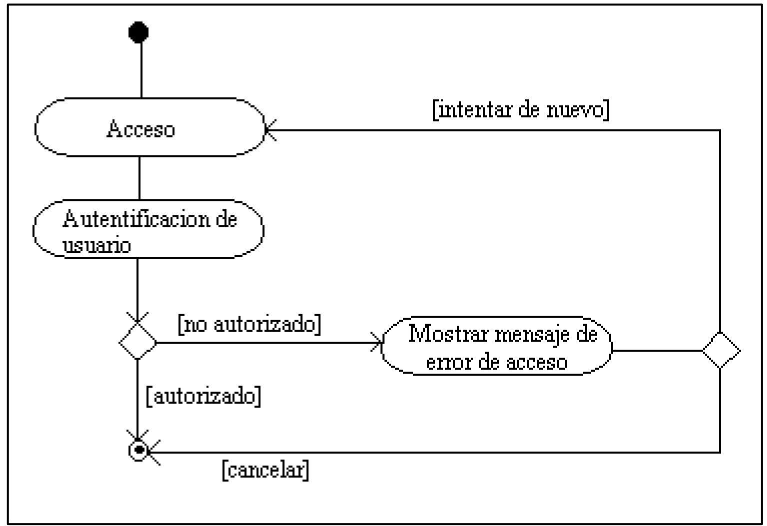


Figura 3.3.1 Diagrama de actividad de autenticación

3.3.2 Consultar estado de cuenta: Recaudador

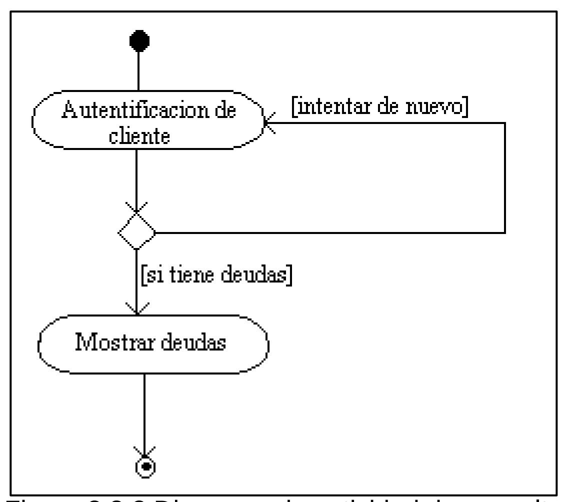


Figura 3.3.2 Diagrama de actividad de consulta estado de cuenta

3.3.3 Apertura de caja: Todos los usuarios

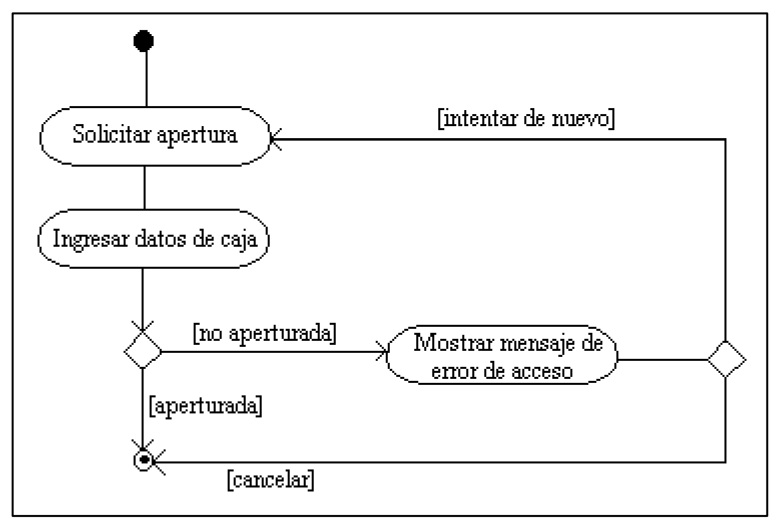


Figura 3.3.3 Diagrama de actividad de apertura de caja

3.3.4 Recaudar valores: Recaudador

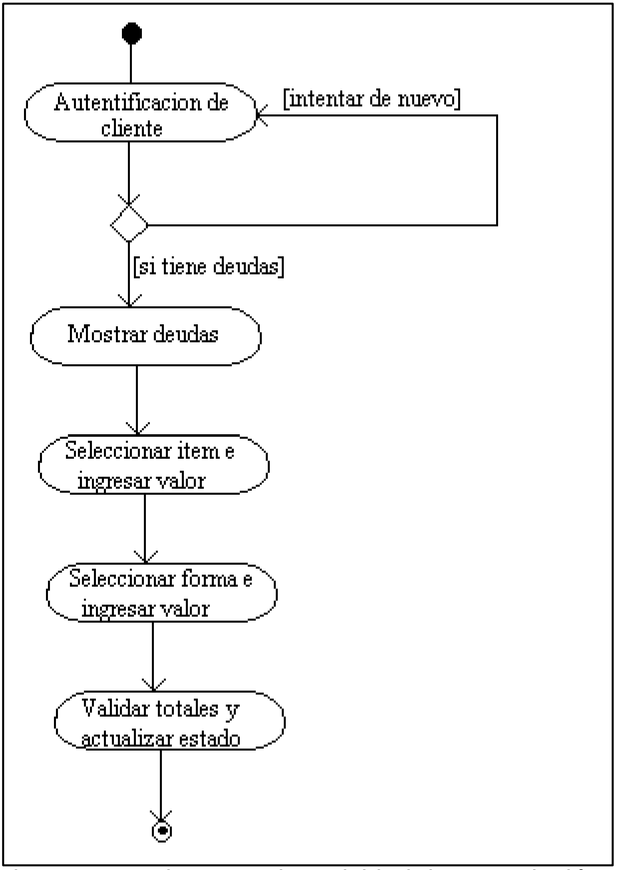


Figura 3.3.4 Diagrama de actividad de recaudación valores

3.3.5 Consultar pagos: Tesorero

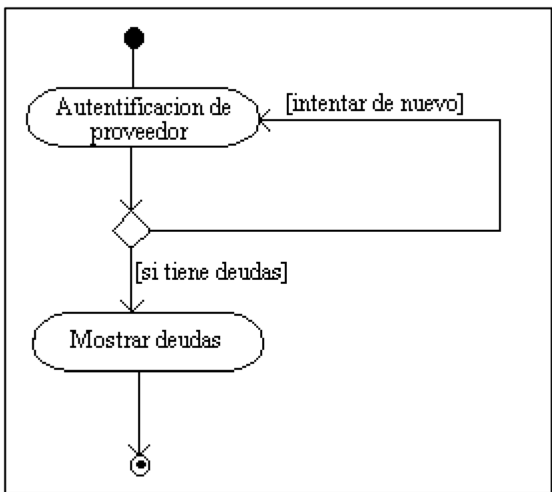


Figura 3.3.5 Diagrama de actividad de consulta pagos

3.3.6 Generación de pagos: Tesorero

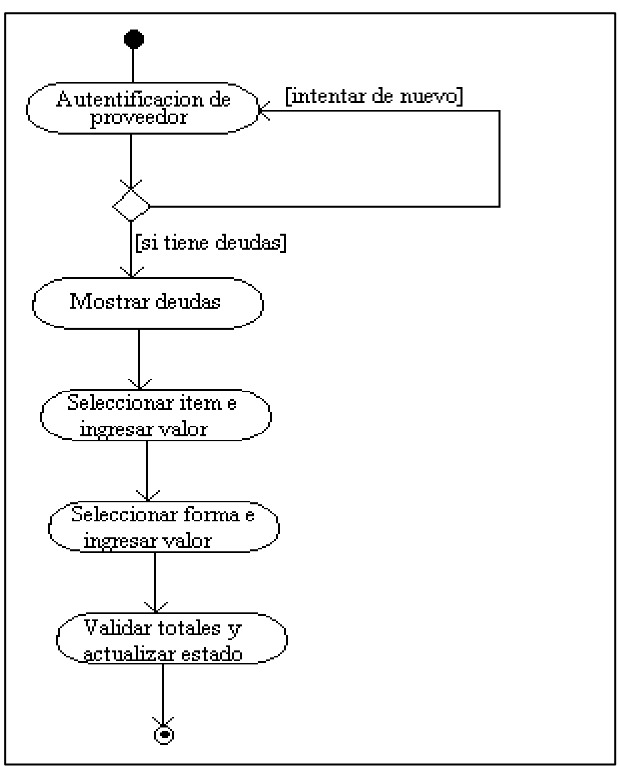


Figura 3.3.6 Diagrama de actividad de generación de pagos

3.3.7 Girar cheque: Tesorero

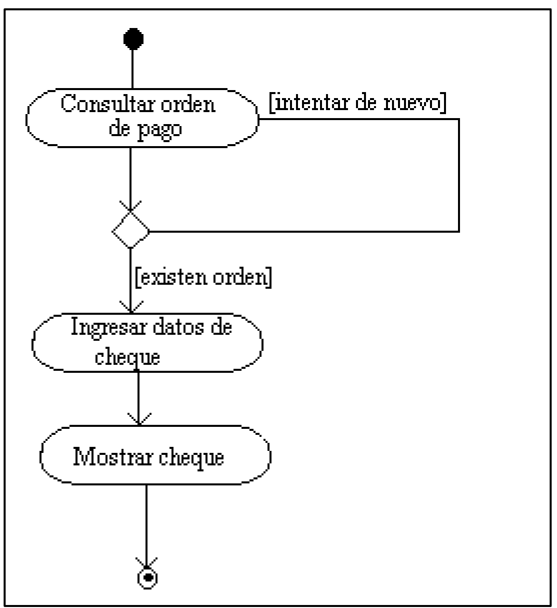


Figura 3.3.7 Diagrama de actividad de girar cheque

3.3.8 Entrega de valores: Pagador

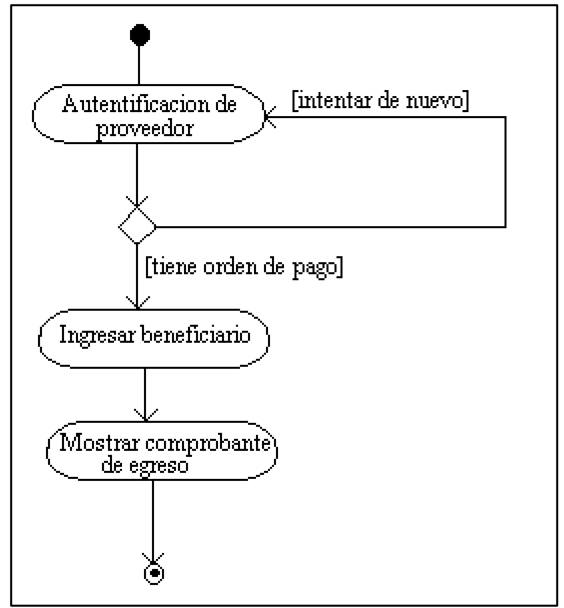


Figura 3.3.8 Diagrama de actividad de entrega de valores

3.3.9 Anular cheque: Tesorero

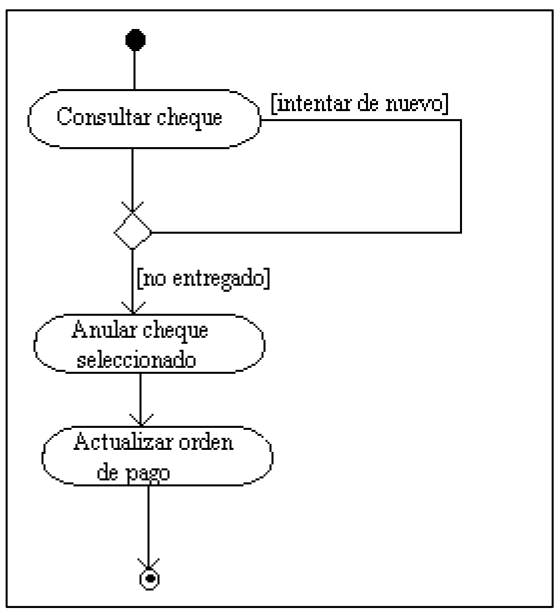


Figura 3.3.9 Diagrama de actividad de anular cheque

3.3.10 Arqueo de caja: Todos los usuarios

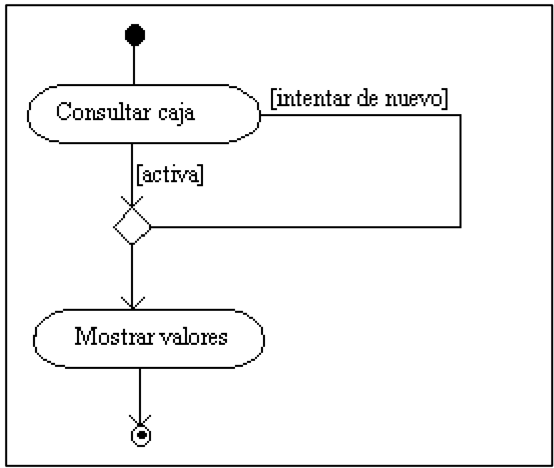


Figura 3.3.10 Diagrama de actividad de arqueo de caja

3.3.11 Cierre de caja: Todos los usuarios

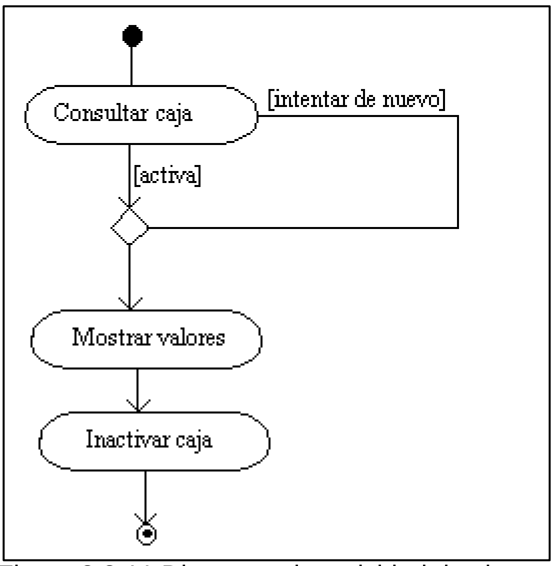


Figura 3.3.11 Diagrama de actividad de cierre de caja

3.3.12 Depositar valores: Tesorero

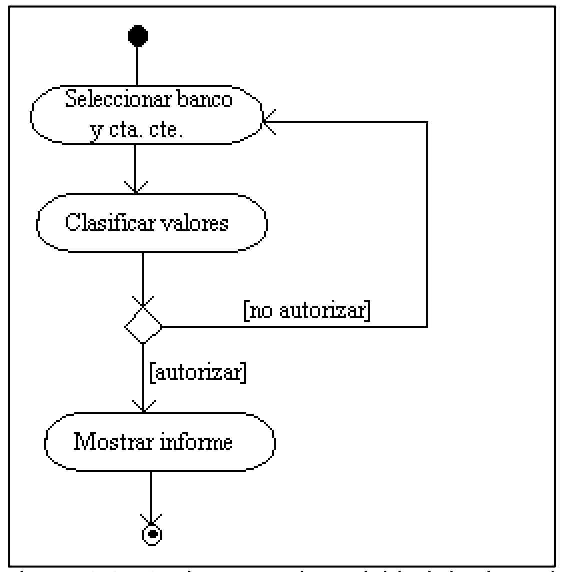


Figura 3.3.12 Diagrama de actividad de depositar valores

3.4 Prototipo de la interfaz

3.4.1 Autenticación



Figura 3.4.1 Prototipo para autenticación

3.4.2 Comprobantes

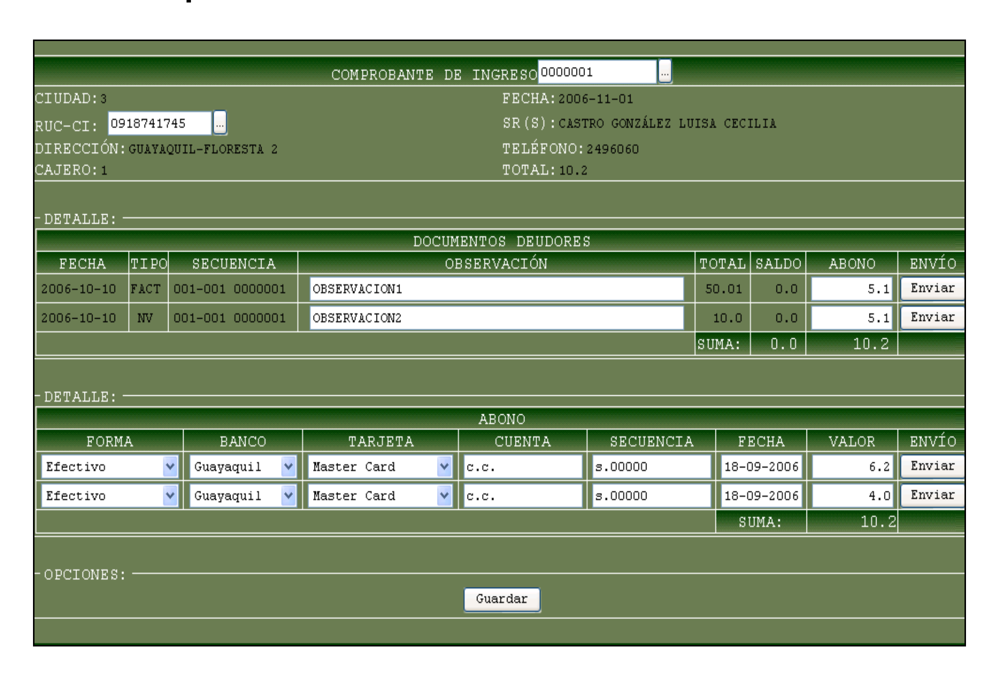


Figura 3.4.2 Prototipo de comprobantes para pagos y cobros

3.4.3 Consultas

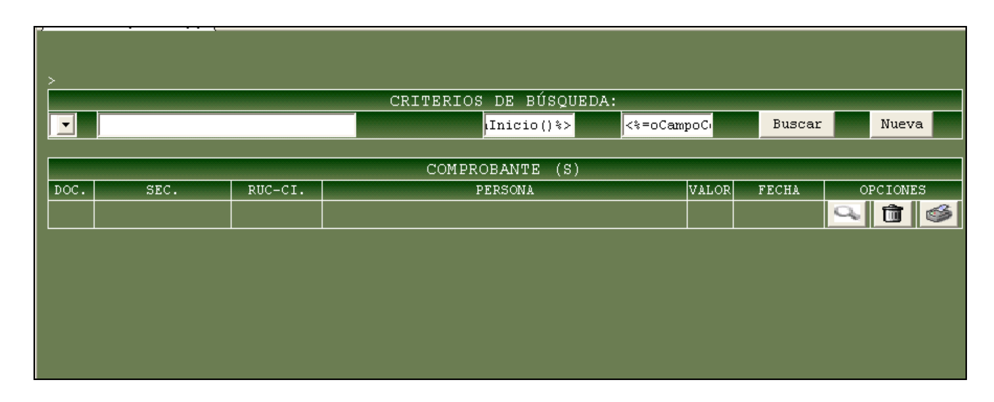


Figura 3.4.3 Prototipo para las consultas

CAPÍTULO 4

4 IMPLEMENTACIÓN

4.1 Modelo

4.1.1 Scripts de base de datos

Script para la creación de la base de datos:

```
CREATE DATABASE "TESORERIA"
  WITH OWNER = postgres
       ENCODING = 'UNICODE'
       TABLESPACE = pg_default;
```

Script para la creación de la tabla “Comprobante”:

```
CREATE TABLE "Comprobante"(
  "Id" int8 NOT NULL,
  "IdDocumento" int8 NOT NULL,
  "IdSucursal" int8 NOT NULL,
  "Secuencia" varchar(15) NOT NULL,
  "IdPersona" int8 NOT NULL,
  "Valor" float8 NOT NULL,
  "IdCajaTurno" int8 NOT NULL,
  "IdEstado" int8 NOT NULL,
  "FechaRegistro" varchar(10) NOT NULL,
  "HoraRegistro" varchar(8) NOT NULL,
  "Beneficiario" varchar(50),
  CONSTRAINT "Comprobante_pkey" PRIMARY KEY ("Id")
)
WITHOUT OIDS;
ALTER TABLE "Comprobante" OWNER TO postgres;
```

Script para la creación de la tabla “ComprobanteItem”:

```
CREATE TABLE "ComprobanteItem"
(
  "Id" int8 NOT NULL,
  "IdComprobante" int8 NOT NULL,
  "IdBitacora" int8 NOT NULL,
  "Saldo" float8 NOT NULL,
  "Abono" float8 NOT NULL,
  "Observacion" varchar(100),
  CONSTRAINT "ComprobanteItem_pkey1" PRIMARY KEY
("IdComprobanteItem"),
  CONSTRAINT "ComprobanteItem_IdBitacora_fkey" FOREIGN KEY
("IdBitacora") REFERENCES "Bitacora" ("IdBitacora") ON UPDATE
RESTRICT ON DELETE RESTRICT
)
WITHOUT OIDS;
ALTER TABLE "ComprobanteItem" OWNER TO postgres;
```

Script para la creación de la tabla “Abono”:

```
CREATE TABLE "Abono"
(
  "Id" int8 NOT NULL,
  "IdComprobante" int8 NOT NULL,
  "IdCatalogoValor" int8 NOT NULL,
  "IdBanco" int8 NOT NULL,
  "IdTarjetaCredito" int8 NOT NULL,
  "CuentaCorriente" varchar(15) NOT NULL,
  "Secuencia" varchar(15) NOT NULL,
  "FechaPlazo" varchar(10) NOT NULL,
  "Valor" float8 NOT NULL,
```

```
  "IdBitacora" int8 NOT NULL,
  "IdEstado" int8 NOT NULL,
  CONSTRAINT "Cobro_pkey" PRIMARY KEY ("IdAbono"),
  CONSTRAINT "Abono_IdBanco_fkey" FOREIGN KEY ("IdBanco")
REFERENCES "Banco" ("IdBanco") ON UPDATE RESTRICT ON DELETE
RESTRICT,
  CONSTRAINT "Abono_IdComprobante_fkey" FOREIGN KEY
("IdComprobante") REFERENCES "Comprobante" ("Id") ON UPDATE
RESTRICT ON DELETE RESTRICT,
  CONSTRAINT "Abono_IdEstado_fkey" FOREIGN KEY ("IdEstado")
REFERENCES "Estado" ("IdEstado") ON UPDATE RESTRICT ON DELETE
RESTRICT,
  CONSTRAINT "Abono_IdTarjetaCredito_fkey" FOREIGN KEY
("IdTarjetaCredito") REFERENCES "TarjetaCredito"
("IdTarjetaCredito") ON UPDATE RESTRICT ON DELETE RESTRICT,
  CONSTRAINT "Cobro_IdBitacoraCobro_fkey" FOREIGN KEY
("IdBitacora") REFERENCES "Bitacora" ("IdBitacora") ON UPDATE
RESTRICT ON DELETE RESTRICT
)
WITHOUT OIDS;
ALTER TABLE "Abono" OWNER TO postgres;
```

4.1.2 Entidades

Clase que modela la tabla “Comprobante”:

```
package com.cisc.erp.be.tesoreria;
import java.util.ArrayList;

public class Comprobante {
    private int IdComprobante;
    private String IdTipoDocumento;
    private String nombre;
    private int IdSucursal;
    private String Secuencia;
    private int IdPersona;
    private String Beneficiario;
    private float Valor;
    private float ValorAbono;
    private int IdCajaTurno;
    private int IdEstado;
    private String FechaRegistro;
    private String HoraRegistro;
    private ArrayList Items;
    private ArrayList Abonos;

    public String getBeneficiario() {
         return Beneficiario;
    }

    public void setBeneficiario(String beneficiario) {
         Beneficiario = beneficiario;
    }
```

```
public String getNombre() {
      if (IdTipoDocumento.equals("CI")){
            this.nombre="COMPROBANTE DE INGRESO: ";
      }

      if (IdTipoDocumento.equals("OPG")){
            this.nombre="ORDEN DE PAGO: ";
      }

      if (IdTipoDocumento.equals("CE")){
            this.nombre="COMPROBANTE DE EGRESO: ";
      }

      return this.nombre;
}

public int getIdComprobante() {
      return IdComprobante;
}

public void setIdComprobante(int idComprobante) {
      IdComprobante = idComprobante;
}

public String getIdTipoDocumento() {
      return IdTipoDocumento;
}

public void setIdTipoDocumento(String idTipoDocumento) {
      IdTipoDocumento = idTipoDocumento;
}

public int getIdSucursal() {
      return IdSucursal;
}

public void setIdSucursal(int idSucursal) {
      IdSucursal = idSucursal;
}

public String getSecuencia() {
      return Secuencia;
}

public void setSecuencia(String secuencia) {
      Secuencia = secuencia;
}

public int getIdPersona() {
      return IdPersona;
}

public void setIdPersona(int idPersona) {
      IdPersona = idPersona;
}

public int getIdEstado() {
```

```
        return IdEstado;
    }

    public void setIdEstado(int idEstado) {
        IdEstado = idEstado;
    }

    public int getIdCajaTurno() {
        return IdCajaTurno;
    }

    public void setIdCajaTurno(int idCajaTurno) {
        IdCajaTurno = idCajaTurno;
    }

    public float getValor() {
        this.Valor = 0.0f;

        for (int i = 0; i < this.Items.size(); i++) {
            ComprobanteItem    oItem    =    (ComprobanteItem)
this.Items.get(i);
            this.Valor = this.Valor + oItem.getAbono();
        }
        return this.Valor;
    }

    public float getValorAbono() {
        this.ValorAbono = 0.0f;

        for (int i = 0; i < this.Abonos.size(); i++) {
            Abono oItem = (Abono) this.Abonos.get(i);
            this.ValorAbono      =      this.ValorAbono      +
oItem.getValor();
        }
        return this.ValorAbono;
    }

    public String getFechaRegistro() {
        return FechaRegistro;
    }

    public void setFechaRegistro(String fechaRegistro) {
        FechaRegistro = fechaRegistro;
    }

    public String getHoraRegistro() {
        return HoraRegistro;
    }

    public void setHoraRegistro(String horaRegistro) {
        HoraRegistro = horaRegistro;
    }

    public String informacionItem() {
        StringBuffer oStringBuffer = new StringBuffer();

        for (int i = 0; i < this.Items.size(); i++) {
```

```
                ComprobanteItem    oItem    =    (ComprobanteItem)
this.Items.get(i);

                if (oItem.getAbono() > 0.0f){

    oStringBuffer.append(oItem.getIdComprobanteItem()  +  "|"  +
oItem.getIdComprobante() + "|" + oItem.getIdBitacora() + "|" +
oItem.getSaldo()   +   "|"   +   oItem.getAbono()   +   "|"   +
oItem.getObservacion() + "#");
                }
          }
          return oStringBuffer.toString();
    }

    public String informacionAbono() {
          StringBuffer oStringBuffer = new StringBuffer();
          for (int i = 0; i < this.Abonos.size(); i++) {
                Abono oItem = (Abono) this.Abonos.get(i);

                if (oItem.getValor() > 0.0f){
                      oStringBuffer.append(oItem.getIdAbono() + "|"
+ oItem.getIdComprobante() + "|" + oItem.getIdCatalogoValor() +
"|" + oItem.getIdBanco() + "|" + oItem.getIdTarjetaCredito() +
"|" + oItem.getCuentaCorriente() + "|" + oItem.getSecuencia() +
"|" + oItem.getFechaPlazo() + "|" + oItem.getValor() + "|" +
oItem.getIdBitacora() + "|" + oItem.getIdEstado() + "#");
                }
          }
          return oStringBuffer.toString();
    }

    public ArrayList getItems() {
          return this.Items;
    }

    public ArrayList getAbonos() {
          return this.Abonos;
    }

    public void agregarItem(ComprobanteItem oComprobanteItem){
          for(int i = 0; i < this.Items.size(); i++){
                ComprobanteItem              oItem              =
(ComprobanteItem)this.Items.get(i);

                if(oItem.getIdBitacora()                        ==
oComprobanteItem.getIdBitacora()){

    oItem.setObservacion(oComprobanteItem.getObservacion());
                      oItem.setAbono(oComprobanteItem.getAbono());
                      return;
                }
          }
          this.Items.add(oComprobanteItem);
    }

    public void agregarAbono(Abono oAbono){
          for(int i = 0; i < this.Abonos.size(); i++){
```

```
              Abono oItem = (Abono)this.Abonos.get(i);

              if(oItem.getIdentificador()                    ==
oAbono.getIdentificador()){

    oItem.setIdCatalogoValor(oAbono.getIdCatalogoValor());
                   oItem.setIdBanco(oAbono.getIdBanco());

    oItem.setIdTarjetaCredito(oAbono.getIdTarjetaCredito());

    oItem.setCuentaCorriente(oAbono.getCuentaCorriente());
                   oItem.setSecuencia(oAbono.getSecuencia());
                   oItem.setFechaPlazo(oAbono.getFechaPlazo());
                   oItem.setValor(oAbono.getValor());
                   return;
              }
         }
         this.Abonos.add(oAbono);
    }

    public void quitarItem(int IdBitacora){
         for(int i = 0; i < this.Items.size(); i++){
              ComprobanteItem              oItem              =
(ComprobanteItem)this.Items.get(i);

              if(oItem.getIdBitacora() == IdBitacora){
                   this.Items.remove(oItem);
                   return;
              }
         }
    }

    public void quitarAbono(int identificador){
         for(int i = 0; i < this.Abonos.size(); i++){
              Abono oItem = (Abono)this.Abonos.get(i);

              if(oItem.getIdentificador() == identificador){
                   this.Abonos.remove(oItem);
                   return;
              }
         }
    }

    public void agregarAbonoFinal(Abono oAbono){
         if  ((IdComprobante  ==  0)  &&  (Abonos.size()-1  ==
oAbono.getIdentificador()) && (oAbono.getValor() > 0.0f)) {
              Abono otroCobro = new Abono();

    otroCobro.setIdentificador(oAbono.getIdentificador() + 1);
              agregarAbono(otroCobro);
         }
    }
         public Comprobante(){
         this.IdComprobante = 0;
         this.IdTipoDocumento = "";
         this.IdSucursal = 0;
         this.Secuencia = "0000000";
```

```
        this.IdPersona = 0;
        this.Beneficiario = "";
        this.IdCajaTurno = 0;
        this.Valor = 0;
        this.ValorAbono = 0;
        this.IdEstado = 0;
        this.FechaRegistro              =              new
java.text.SimpleDateFormat("yyyy-MM-dd").format(new
java.util.Date());
        this.HoraRegistro = "00:00:00";
        this.Items = new ArrayList();
        this.Abonos = new ArrayList();
    }

    public      Comprobante(int      IdComprobante,      String
IdTipoDocumento,   int   IdSucursal,   String   Secuencia,   int
IdPersona,  String  beneficiario,  int  IdCajaTurno,  float  Valor,
float  ValorAbono,  int  IdEstado,  String  FechaRegistro,  String
HoraRegistro) {
        this.IdComprobante = IdComprobante;
        this.IdTipoDocumento = IdTipoDocumento;
        this.IdSucursal = IdSucursal;
        this.Secuencia = Secuencia;
        this.IdPersona = IdPersona;
        this.Beneficiario = beneficiario;
        this.IdCajaTurno = IdCajaTurno;
        this.Valor = Valor;
        this.ValorAbono = ValorAbono;
        this.IdEstado = IdEstado;
        this.FechaRegistro = FechaRegistro;
        this.HoraRegistro = HoraRegistro;
        this.Items = new ArrayList();
        this.Abonos = new ArrayList();
    }
}
```

Clase que modela la tabla “ComprobanteItem”:

```
package com.cisc.erp.be.tesoreria;

public class ComprobanteItem {
    private int IdComprobanteItem;
    private int IdComprobante;
    private int IdBitacora;
    private float Saldo;
    private float Abono;
    private String Observacion;

    public float getSaldo() {
        return Saldo;
    }

    public void setSaldo(float saldo) {
        Saldo = saldo;
    }
```

```
    public int getIdComprobanteItem() {
          return IdComprobanteItem;
    }

    public void setIdComprobanteItem(int idComprobanteItem) {
          IdComprobanteItem = idComprobanteItem;
    }

    public int getIdComprobante() {
          return IdComprobante;
    }

    public void setIdComprobante(int idComprobante) {
          IdComprobante = idComprobante;
    }

    public int getIdBitacora() {
          return IdBitacora;
    }

    public void setIdBitacora(int idBitacora) {
          IdBitacora = idBitacora;
    }

    public float getAbono() {
          return Abono;
    }

    public void setAbono(float abono) {
          Abono = abono;
    }

    public String getObservacion() {
          return Observacion;
    }

    public void setObservacion(String observacion) {
          Observacion = observacion;
    }

    public ComprobanteItem(){
          this.IdComprobanteItem = 0;
          this.IdComprobante = 0;
          this.IdBitacora = 0;
          this.Saldo = 0;
          this.Abono = 0;
          this.Observacion = "Ninguna.";
    }

    public        ComprobanteItem(int        IdComprobanteItem,int
IdComprobante,int  IdBitacora,  float  Saldo,  float  Abono,  String
Observacion){
          this.IdComprobanteItem = IdComprobanteItem;
          this.IdComprobante = IdComprobante;
          this.IdBitacora = IdBitacora;
          this.Saldo = Saldo;
```

```
        this.Abono = Abono;
        this.Observacion = Observacion;
    } }
```

Clase que modela la tabla “Abono”:

```
package com.cisc.erp.be.tesoreria;

public class Abono {
     private int IdAbono;
     private int IdComprobante;
     private int Identificador;
     private int IdCatalogoValor;
     private int IdBanco;
     private int IdTarjetaCredito;
     private String CuentaCorriente;
     private String Secuencia;
     private String FechaPlazo;
     private float Valor;
     private int IdBitacora;
     private int IdEstado;

    public int getIdEstado() {
         return IdEstado;
    }

    public int getIdBitacora() {
         return IdBitacora;
    }

    public void setIdEstado(int idEstado) {
         IdEstado = idEstado;
    }

    public void setIdBitacora(int idBitacoraCobro) {
         IdBitacora = idBitacoraCobro;
    }

    public int getIdAbono() {
         return IdAbono;
    }

    public int getIdComprobante() {
         return IdComprobante;
    }

    public int getIdentificador() {
         return Identificador;
    }

    public void setIdentificador(int identificador) {
         Identificador = identificador;
    }

    public int getIdCatalogoValor() {
```

```
        return IdCatalogoValor;
    }

    public int getIdBanco() {
        return IdBanco;
    }

    public int getIdTarjetaCredito() {
        return IdTarjetaCredito;
    }

    public String getCuentaCorriente() {
        return CuentaCorriente;
    }

    public String getSecuencia() {
        return Secuencia;
    }

    public String getFechaPlazo() {
        return FechaPlazo;
    }

    public float getValor() {
        return Valor;
    }

    public void setCuentaCorriente(String cuentaCorriente) {
        CuentaCorriente = cuentaCorriente.trim();
    }

    public void setFechaPlazo(String fechaPlazo) {
        FechaPlazo = fechaPlazo.trim();
    }

    public void setIdBanco(int idBanco) {
        IdBanco = idBanco;
    }

    public void setIdCatalogoValor(int idCatalogoValor) {
        IdCatalogoValor = idCatalogoValor;
    }

    public void setIdAbono(int idCobro) {
        IdAbono = idCobro;
    }

    public void setIdComprobante(int idComprobanteIngreso) {
        IdComprobante = idComprobanteIngreso;
    }

    public void setIdTarjetaCredito(int idTarjetaCredito) {
        IdTarjetaCredito = idTarjetaCredito;
    }

    public void setSecuencia(String secuencia) {
        Secuencia = secuencia.trim();
```

```
    }

    public void setValor(float valor) {
         Valor = valor;
    }

    public Abono(){
          this.IdAbono = 0;
          this.IdComprobante = 0;
          this.Identificador = 0;
          this.IdCatalogoValor = 2;
          this.IdBanco = 0;
          this.IdTarjetaCredito = 0;
          this.CuentaCorriente = "";
          this.Secuencia = "";
          this.FechaPlazo                  =                 new
java.text.SimpleDateFormat("yyyy-MM-dd").format(new
java.util.Date());
          this.Valor = 0.0f;
          this.IdBitacora = 0;
          this.IdEstado = 1;
    }

    public  Abono(int  idCobro,  int  idComprobanteIngreso,  int
identificador,   int   idCatalogoValor,   int   idBanco,   int
idTarjetaCredito,  String  cuentaCorriente,  String  secuencia,
String  fechaPlazo,  float  valor,  int  idBitacoraCobro,  int
idEstado) {
          this.IdAbono = idCobro;
          this.IdComprobante = idComprobanteIngreso;
          this.Identificador = identificador;
          this.IdCatalogoValor = idCatalogoValor;
          this.IdBanco = idBanco;
          this.IdTarjetaCredito = idTarjetaCredito;
          this.CuentaCorriente = cuentaCorriente.trim();
          this.Secuencia = secuencia.trim();
          this.FechaPlazo = fechaPlazo.trim();
          this.Valor = valor;
          this.IdBitacora = idBitacoraCobro;
          this.IdEstado = idEstado;
    }
}
```

4.1.3 Conexión a base de datos

Clase que contiene la conexión a la base de datos:

```
package com.cisc.erp.dao.tesoreria;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Conexion {
```

```
    private static String error;
    private static Connection conexion = null;

    public Conexion(){
         super();
    }

    public    static    Connection    getConnection()    throws
ClassNotFoundException, SQLException{
    try {
         if ((conexion == null)||(conexion.isClosed())) {
              Class.forName("org.postgresql.Driver");

    //Class.forName("org.postgresql.Driver").newInstance();
              conexion                                         =
DriverManager.getConnection("jdbc:postgresql://SERVIDOR/TESORERI
A","postgres","roca");
              conexion.setAutoCommit(true);
         }
    } catch (ClassNotFoundException cnfe) {
         error = "ClassNotFoundException: Localición fallida del
driver de conexión.";
         throw new ClassNotFoundException(error);
    } catch (SQLException cnfe) {
         error = cnfe.toString();
         throw new SQLException(error);
    }
    return conexion;
    }

    public   static   Connection   getConnection(String   usuario,
String contrasena) throws ClassNotFoundException, SQLException {
         try{
              if((conexion == null)||(conexion.isClosed())){
                   Class.forName("org.postgresql.Driver");
                   conexion                                    =
DriverManager.getConnection("jdbc:postgresql://localhost/TESORER
IA", usuario, contrasena);
                   conexion.setAutoCommit(true);
              }
         } catch (ClassNotFoundException cnfe) {
              error    =    "ClassNotFoundException:    Localición
fallida del driver de conexión.";
              throw new ClassNotFoundException(error);
         } catch (SQLException cnfe) {
              error = cnfe.toString();
              throw new SQLException(error);
         }
         return conexion;
         }

    public static void disconnect() throws SQLException {
         try {
              if ( conexion != null ) {
                   conexion.close();
              }
         }
```

```
        catch (SQLException sqle) {
              error  =  ("SQLException:  Unable  to  close  the
database connection.");
              throw new SQLException(error);
        }
    }     }
```

4.1.4 Excepciones

Clase que maneja las excepciones:

```
package com.cisc.erp.exception.tesoreria;

public class TesoreriaException extends Exception {

    private String usuario = "null";
    private String mensaje = "";

    public String toString () {
          return (super.toString() + mensaje);
    }

    public TesoreriaException () {}

    public TesoreriaException (String msg) {
          super(msg);
          mensaje = "";
    }

    public TesoreriaException (String msg, String usuario) {
          super(msg);
          this.setUsuario(usuario);
          mensaje = ": The user is " + this.getUsuario();
    }

    public void setUsuario (String value){
          if (value == null){
                value = "null";
          }
          usuario = value;
    }

    public String getUsuario (){
          return usuario;
    }

}
```

4.1.5 Objetos de acceso a datos

Clase que maneja el acceso a datos para los comprobantes:

```
package com.cisc.erp.dao.tesoreria;
import java.sql.*;
import java.util.ArrayList;

import com.cisc.erp.be.tesoreria.*;
import com.cisc.erp.exception.tesoreria.TesoreriaException;


public class ComprobanteDao {
    private Connection con = null;

    public ComprobanteDao(Connection con) {
         this.con = con;
    }

    public    void    insertar(Comprobante    oComprobante)throws
SQLException, Exception {
  if (con != null){
      PreparedStatement oPreparedStatement;
    oPreparedStatement       =       con.prepareStatement("SELECT
\"insertarComprobante\" (?,?,?,?,?,?,?,?,?);");
    oPreparedStatement.setString(1,
oComprobante.getIdTipoDocumento());
    oPreparedStatement.setInt(2, oComprobante.getIdSucursal());
    oPreparedStatement.setString(3,
oComprobante.getSecuencia());
    oPreparedStatement.setInt(4, oComprobante.getIdPersona());
              oPreparedStatement.setString(5,
oComprobante.getBeneficiario());
              oPreparedStatement.setInt(6,
oComprobante.getIdCajaTurno());
              oPreparedStatement.setFloat(7,
oComprobante.getValor());
              oPreparedStatement.setString(8,
oComprobante.informacionItem());
              oPreparedStatement.setString(9,
oComprobante.informacionAbono());
              oPreparedStatement.execute();
         }else{
              throw    new    TesoreriaException    ("Exception:
Connection to database was lost.");
         }
    }

    public  Comprobante  seleccionar(String  idComprobante)  throws
SQLException, Exception {
         Comprobante oComprobante = new Comprobante();
         ResultSet rs = null;
         ResultSet rs2 = null;
         String     queryString     =     ("SELECT     *     FROM
\"seleccionarComprobante\"('" + idComprobante + "');");
         Statement stmt = con.createStatement();
         rs = stmt.executeQuery(queryString);
         while (rs.next()) {

oComprobante.setIdComprobante(rs.getInt("IdComprobante"));
```

```
oComprobante.setIdTipoDocumento(rs.getString("IdTipoDocumento"))
;

oComprobante.setIdSucursal(rs.getInt("IdSucursal"));

oComprobante.setSecuencia(rs.getString("Secuencia"));


oComprobante.setIdPersona(rs.getInt("IdPersona"));

oComprobante.setBeneficiario(rs.getString("Beneficiario"));

oComprobante.setIdCajaTurno(rs.getInt("IdCajaTurno"));
               oComprobante.setIdEstado(rs.getInt("IdEstado"));

oComprobante.setFechaRegistro(rs.getString("FechaRegistro"));

oComprobante.setHoraRegistro(rs.getString("HoraRegistro"));

      queryString        =         ("SELECT        *        FROM
\"seleccionarComprobanteItem\"('"                              +
oComprobante.getIdComprobante() + "');");
   stmt = con.createStatement();
   rs2 = stmt.executeQuery(queryString);
   while (rs2.next()) {
    ComprobanteItem oComprobanteItem = new ComprobanteItem();


oComprobanteItem.setIdComprobanteItem(rs2.getInt("IdComprobanteI
tem"));

oComprobanteItem.setIdComprobante(rs2.getInt("IdComprobante"));

oComprobanteItem.setIdBitacora(rs2.getInt("IdBitacora"));

oComprobanteItem.setSaldo(rs2.getFloat("Saldo"));

oComprobanteItem.setAbono(rs2.getFloat("Abono"));

oComprobanteItem.setObservacion(rs2.getString("Observacion"));
                oComprobante.agregarItem(oComprobanteItem);
               }
               rs2.close();

               queryString      =      ("SELECT      *      FROM
\"seleccionarAbono\"("   +   oComprobante.getIdComprobante()   +
");");
               stmt = con.createStatement();
               rs2 = stmt.executeQuery(queryString);
               int i = 1;
               while (rs2.next()) {
                Abono oCobro = new Abono();
                oCobro.setIdAbono(rs2.getInt("IdAbono"));

oCobro.setIdComprobante(rs2.getInt("IdComprobante"));
                oCobro.setIdentificador(i);
```

```
oCobro.setIdCatalogoValor(rs2.getInt("IdCatalogoValor"));
                oCobro.setIdBanco(rs2.getInt("IdBanco"));

oCobro.setIdTarjetaCredito(rs2.getInt("IdTarjetaCredito"));

oCobro.setCuentaCorriente(rs2.getString("CuentaCorriente"));
                oCobro.setSecuencia(rs2.getString("Secuencia"));

oCobro.setFechaPlazo(rs2.getString("FechaPlazo"));
                oCobro.setValor(rs2.getFloat("Valor"));
                oCobro.setIdBitacora(rs2.getInt("IdBitacora"));
                oCobro.setIdEstado(rs2.getInt("IdEstado"));
                oComprobante.agregarAbono(oCobro);
                i++;
               }
               rs2.close();
          }
          rs.close();
          return oComprobante;
    }

    public  void  actualizar(Comprobante  oComprobante)  throws
SQLException, Exception{
         if (con != null){
              PreparedStatement oPreparedStatement;
              oPreparedStatement  =  con.prepareStatement("SELECT
\"actualizarComprobante\"(?,?,?,?,?,?,?,?,?,?);");
oPreparedStatement.setInt(1, oComprobante.getIdComprobante());
              oPreparedStatement.setString(2,
oComprobante.getIdTipoDocumento());
              oPreparedStatement.setInt(3,
oComprobante.getIdSucursal());
              oPreparedStatement.setString(4,
oComprobante.getSecuencia());
              oPreparedStatement.setInt(5,
oComprobante.getIdPersona());
              oPreparedStatement.setString(6,
oComprobante.getBeneficiario());
              oPreparedStatement.setFloat(7,
oComprobante.getValor());
              oPreparedStatement.setInt(8,
oComprobante.getIdEstado());
              oPreparedStatement.setString(9,
oComprobante.informacionItem());
              oPreparedStatement.setString(10,
oComprobante.informacionAbono());
              oPreparedStatement.execute();
         }else{
              throw    new    TesoreriaException    ("Exception:
Connection to database was lost.");
         }
    }

    public void eliminar(int idComprobante) throws SQLException,
Exception{
         if (con != null){
```

```
                PreparedStatement oPreparedStatement;
                oPreparedStatement  =  con.prepareStatement("SELECT
\"eliminarComprobante\"(?);");
                oPreparedStatement.setInt(1, idComprobante);
                oPreparedStatement.execute();
          }else{
                throw    new    TesoreriaException    ("Exception:
Connection to database was lost.");}
    }

    public   ArrayList   busqueda(String   tipoDocumento,   String
secuencia,  String  fechaInicio,  String  fechaFin,  String  rucCi,
String apellidos, String nombres) throws SQLException, Exception
{
          ArrayList comprobantes = new ArrayList();
          ResultSet rs = null;
          ResultSet rs2 = null;
          String     queryString     =     ("SELECT     *     FROM
\"buscarComprobante\"('" + tipoDocumento +  "','" + secuencia +
"','" + fechaInicio + "','" + fechaFin + "','" + rucCi + "','" +
apellidos + "','" + nombres + "');");
          Statement stmt = con.createStatement();
          rs = stmt.executeQuery(queryString);
          while (rs.next()) {
                Comprobante oComprobante = new Comprobante();
oComprobante.setIdComprobante(rs.getInt("IdComprobante"));


    oComprobante.setIdTipoDocumento(rs.getString("IdTipoDocument
o"));

    oComprobante.setIdSucursal(rs.getInt("IdSucursal"));

    oComprobante.setSecuencia(rs.getString("Secuencia"));
                oComprobante.setIdPersona(rs.getInt("IdPersona"));

    oComprobante.setBeneficiario(rs.getString("Beneficiario"));

    oComprobante.setIdCajaTurno(rs.getInt("IdCajaTurno"));
                oComprobante.setIdEstado(rs.getInt("IdEstado"));

    oComprobante.setFechaRegistro(rs.getString("FechaRegistro"))
;

    oComprobante.setHoraRegistro(rs.getString("HoraRegistro"));

                queryString      =      ("SELECT      *      FROM
\"seleccionarComprobanteItem\"('"   +
oComprobante.getIdComprobante() + "');");
                stmt = con.createStatement();
                rs2 = stmt.executeQuery(queryString);
                while (rs2.next()) {
                      ComprobanteItem   oComprobanteItem   =   new
ComprobanteItem();

    oComprobanteItem.setIdComprobanteItem(rs2.getInt("IdComproba
nteItem"));
```

```
    oComprobanteItem.setIdComprobante(rs2.getInt("IdComprobante"
));

    oComprobanteItem.setIdBitacora(rs2.getInt("IdBitacora"));

    oComprobanteItem.setSaldo(rs2.getFloat("Saldo"));
    oComprobanteItem.setAbono(rs2.getFloat("Abono"));

    oComprobanteItem.setObservacion(rs2.getString("Observacion")
);
                    oComprobante.agregarItem(oComprobanteItem);
               }
               rs2.close();

               queryString      =      ("SELECT      *      FROM
\"seleccionarAbono\"("+ oComprobante.getIdComprobante() + ");");
               stmt = con.createStatement();
               rs2 = stmt.executeQuery(queryString);
               int i = 1;
               while (rs2.next()) {
                    Abono oCobro = new Abono();
                    oCobro.setIdAbono(rs2.getInt("IdAbono"));

    oCobro.setIdComprobante(rs2.getInt("IdComprobante"));
                    oCobro.setIdentificador(i);
    oCobro.setIdCatalogoValor(rs2.getInt("IdCatalogoValor"));
                    oCobro.setIdBanco(rs2.getInt("IdBanco"));

    oCobro.setIdTarjetaCredito(rs2.getInt("IdTarjetaCredito"));

    oCobro.setCuentaCorriente(rs2.getString("CuentaCorriente"));

    oCobro.setSecuencia(rs2.getString("Secuencia"));

    oCobro.setFechaPlazo(rs2.getString("FechaPlazo"));
                    oCobro.setValor(rs2.getFloat("Valor"));

    oCobro.setIdBitacora(rs2.getInt("IdBitacora"));
                    oCobro.setIdEstado(rs2.getInt("IdEstado"));
                    oComprobante.agregarAbono(oCobro);
                    i++;
               }
               rs2.close();
               comprobantes.add(oComprobante);
          }
          rs.close();
          return comprobantes;
    }
}
```

4.1.6 Funciones de base de datos

Función para insertar un comprobante a la base:

```
CREATE OR REPLACE FUNCTION "insertarComprobante"(idDocumento
int8, idSucursal int8, secuencia2 varchar, idPersona int8,
beneficiario varchar, idCajaTurno int8, valor float8, items
text, abonos text)
RETURNS void AS
$BODY$
DECLARE
 /* Variables para comprobante. */
 idComprobante "Comprobante"."Id"%TYPE;
 idComprobanteItem "ComprobanteItem"."IdComprobanteItem"%TYPE;
 secuencia "Comprobante"."Secuencia"%TYPE;
 idAbono "Abono"."IdAbono"%TYPE;

 /* Variables operativas. */
 texto text;
 delimitador text;
 campo integer;

 /* Variables para documentos afectados. */
 idBitacora "ComprobanteItem"."IdBitacora"%TYPE;
 saldo "ComprobanteItem"."Saldo"%TYPE;
 abono "ComprobanteItem"."Abono"%TYPE;
 observacion "ComprobanteItem"."Observacion"%TYPE;

 /* Variables para cobros. */
 idCatalogoValor "Abono"."IdCatalogoValor"%TYPE;
 idBanco "Abono"."IdBanco"%TYPE;
 idTarjetaCredito "Abono"."IdTarjetaCredito"%TYPE;
 cuentaCorriente "Abono"."CuentaCorriente"%TYPE;
 secuenciaAbono "Abono"."Secuencia"%TYPE;
 fechaPlazo "Abono"."FechaPlazo"%TYPE;
 valorAbono "Abono"."Valor"%TYPE;
 idBitacoraAbono "Abono"."IdBitacora"%TYPE;
BEGIN
  idComprobante:=(SELECT MAX("Id") + 1 FROM "Comprobante");

  IF idComprobante IS NULL THEN
     idComprobante:=1;
  END IF;

  secuencia       :=(SELECT       "generarSecuencia"(idDocumento,
idSucursal));

  INSERT INTO "Comprobante"
  (
   "Id",
   "IdDocumento",
   "IdSucursal",
   "Secuencia",
   "IdPersona",
   "Valor",
   "IdCajaTurno",
   "IdEstado",
   "FechaRegistro",
   "HoraRegistro",
   "Beneficiario"
  )
```

```
  VALUES
  (
   idComprobante,
   idDocumento,
   idSucursal,
   secuencia,
   idPersona,   valor,
   idCajaTurno,
   1,
   CURRENT_DATE,
   LOCALTIME(0),
   beneficiario);

 /* Inserción de los documentos afectados.*/
 campo:=0;
 LOOP
    idComprobanteItem:=(SELECT   MAX("IdComprobanteItem")   FROM
"ComprobanteItem") + 1;

    IF idComprobanteItem IS NULL THEN
       idComprobanteItem:=1;
    END IF;

    campo:= campo + 1;
    delimitador:='#';
    SELECT "Token"(items, delimitador, campo) INTO texto;
    IF texto <> '' THEN
    delimitador:='|';

    SELECT  CAST("Token"(texto,  delimitador,  3)  AS  int8)  INTO
idBitacora;
    SELECT  CAST("Token"(texto,  delimitador,  4)  AS  float8)  INTO
saldo;
    SELECT  CAST("Token"(texto,  delimitador,  5)  AS  float8)  INTO
abono;
    SELECT  CAST("Token"(texto,  delimitador,  6)  AS  varchar)  INTO
observacion;

      INSERT INTO "ComprobanteItem"
      (
       "IdComprobanteItem",
       "IdComprobante",
       "IdBitacora",
     "Saldo",
       "Abono",
       "Observacion"
      )
      VALUES
     (
      idComprobanteItem,
      idComprobante,
      idBitacora,
    saldo,
      abono,
      observacion
     );
    ELSE
```

```
    EXIT;
    END IF;
 END LOOP;

 /* Inserción de los cobros.*/
 campo:=0;
 LOOP
    idAbono:=(SELECT MAX("IdAbono") FROM "Abono") + 1;

    IF idAbono IS NULL THEN
       idAbono:=1;
    END IF;

    campo:= campo + 1;
    delimitador:='#';
    SELECT "Token"(abonos, delimitador, campo) INTO texto;
    IF texto <> '' THEN
    delimitador:='|';

    SELECT  CAST("Token"(texto,  delimitador,  3)  AS  int8)  INTO
idCatalogoValor;
    SELECT  CAST("Token"(texto,  delimitador,  4)  AS  int8)  INTO
idBanco;
    SELECT  CAST("Token"(texto,  delimitador,  5)  AS  int8)  INTO
idTarjetaCredito;
    SELECT CAST("Token"(texto, delimitador, 6) AS varchar) INTO
cuentaCorriente;
    SELECT CAST("Token"(texto, delimitador, 7) AS varchar) INTO
secuenciaAbono;
    SELECT CAST("Token"(texto, delimitador, 8) AS varchar) INTO
fechaPlazo;
    SELECT  CAST("Token"(texto,  delimitador,  9)  AS  float8)  INTO
valorAbono;
      SELECT CAST("Token"(texto, delimitador, 10) AS int8) INTO
idBitacoraAbono;

      INSERT INTO "Abono"
      (
    "IdAbono",
    "IdComprobante",
    "IdCatalogoValor",
    "IdBanco",
    "IdTarjetaCredito",
    "CuentaCorriente",
    "Secuencia",
    "FechaPlazo",
    "Valor",
      "IdBitacora",
      "IdEstado"
      )
      VALUES
     (
    idAbono,
      idComprobante,
    idCatalogoValor,
    idBanco,
    idTarjetaCredito,
```

```
    cuentaCorriente,
    secuenciaAbono,
    fechaPlazo,
    valorAbono,
      idBitacoraAbono,
      1
     );
    ELSE
    EXIT;
    END IF;
 END LOOP;

  RETURN;
END;
$BODY$
LANGUAGE 'plpgsql' VOLATILE;
```

Función para seleccionar un comprobante:

```
CREATE OR REPLACE FUNCTION "seleccionarComprobante"(int8)
  RETURNS SETOF "Comprobante" AS
$BODY$
     SELECT
    "Id",
    "IdDocumento",
    "IdSucursal",
    "Secuencia",
    "IdPersona",
    "Valor",
    "IdCajaTurno",
    "IdEstado",
    "FechaRegistro",
    "HoraRegistro",
    "Beneficiario"
    FROM
    "Comprobante"
    WHERE
    "Id" LIKE $1 AND
    "IdEstado" = 1;
$BODY$
  LANGUAGE 'sql' VOLATILE;
```

Función para seleccionar los items del comprobante:

```
CREATE OR REPLACE FUNCTION "seleccionarComprobanteItem"(int8)
  RETURNS SETOF "ComprobanteItem" AS
$BODY$
     SELECT
    "IdComprobanteItem",
    "IdComprobante",
    "IdBitacora",
    "Saldo",
```

```
    "Abono",
    "Observacion"
    FROM
    "ComprobanteItem"
    WHERE
    "IdComprobante" LIKE $1;
$BODY$
  LANGUAGE 'sql' VOLATILE;
```

Función para seleccionar las formas de pago de un comprobante:

```
CREATE OR REPLACE FUNCTION "seleccionarAbono"(int8)
  RETURNS SETOF "Abono" AS
$BODY$
     SELECT
    "IdAbono",
    "IdComprobante",
    "IdCatalogoValor",
    "IdBanco",
    "IdTarjetaCredito",
    "CuentaCorriente",
    "Secuencia",
    "FechaPlazo",
    "Valor",
        "IdBitacora",
        "IdEstado"
    FROM
    "Abono"
    WHERE
    "IdComprobante" = $1;
$BODY$
  LANGUAGE 'sql' VOLATILE;
```

Función para buscar un comprobante por medio de varios criterios:

```
CREATE   OR   REPLACE   FUNCTION   "buscarComprobante"("varchar",
"varchar",    "varchar",    "varchar",    "varchar",    "varchar",
"varchar")
  RETURNS SETOF "Comprobante" AS
$BODY$
     SELECT
        DISTINCT ON (a."Id")
    a."Id",
    a."IdDocumento",
    a."IdSucursal",
    a."Secuencia",
    a."IdPersona",
    a."Valor",
    a."IdCajaTurno",
```

```
    a."IdEstado",
    a."FechaRegistro",
    a."HoraRegistro",
    a."Beneficiario"
    FROM
    "Comprobante"   a,   "ComprobanteItem"   b,   "Bitacora"   c,
"Documento" d
    WHERE
    a."Id" = b."IdComprobante" AND
    a."IdDocumento" = d."Id" AND
    b."IdBitacora" = c."IdBitacora" AND
    UPPER(d."Sigla") LIKE LTRIM(UPPER($1)) AND
    a."Secuencia" LIKE '%' || LTRIM($2) || '%' AND
    a."FechaRegistro" BETWEEN CAST ($3 AS Date ) AND CAST ($4 AS
Date ) AND
    c."RucCi" LIKE '%' || LTRIM($5) || '%' AND
    UPPER(c."Apellidos") LIKE '%' || LTRIM(UPPER($6)) || '%' AND
    (UPPER(c."Nombres") LIKE '%' || LTRIM(UPPER($7)) || '%' OR
c."Nombres" IS NULL);
$BODY$
  LANGUAGE 'sql' VOLATILE;
```

Función para actualizar un comprobante:

```
CREATE OR REPLACE FUNCTION "actualizarComprobante"(idComprobante
int8,  idDocumento  int8,  idSucursal  int8,  secuencia  varchar,
idPersona  int8,  beneficiario  varchar,  valor  float8,  idEstado
int8, items text, abonos text)
RETURNS void AS
$BODY$
DECLARE
 /* Variables operativas. */
 texto text;
 delimitador text;
 campo integer;

 /* Variables para documentos afectados. */
 idComprobanteItem "ComprobanteItem"."IdComprobanteItem"%TYPE;
 idBitacora "ComprobanteItem"."IdBitacora"%TYPE;
 saldo "ComprobanteItem"."Saldo"%TYPE;
 abono "ComprobanteItem"."Abono"%TYPE;
 observacion "ComprobanteItem"."Observacion"%TYPE;

 /* Variables para cobros. */
 idAbono "Abono"."IdAbono"%TYPE;
 idCatalogoValor "Abono"."IdCatalogoValor"%TYPE;
 idBanco "Abono"."IdBanco"%TYPE;
 idTarjetaCredito "Abono"."IdTarjetaCredito"%TYPE;
 cuentaCorriente "Abono"."CuentaCorriente"%TYPE;
 secuenciaAbono "Abono"."Secuencia"%TYPE;
 fechaPlazo "Abono"."FechaPlazo"%TYPE;
 valorAbono "Abono"."Valor"%TYPE;
 idBitacoraAbono "Abono"."IdBitacora"%TYPE;
 idEstadoAbono "Abono"."IdEstado"%TYPE;
BEGIN
```

```
 /* Actualización de comprobante. */
  UPDATE
    "Comprobante"
  SET
    "IdDocumento" = idDocumento,
    "IdSucursal" = idSucursal,
    "Secuencia" = secuencia,
    "IdPersona" = idPersona,
    "Valor" = valor,
    "IdEstado" = idEstado,
    "Beneficiario" = beneficiario
  WHERE
    "Id" = idComprobante;

 /* Actualización de los documentos afectados. */
  campo:=0;
  LOOP
     campo:= campo + 1;
     delimitador:='#';
     SELECT "Token"(items, delimitador, campo) INTO texto;
     IF texto <> '' THEN
     delimitador:='|';

     SELECT  CAST("Token"(texto,  delimitador,  1)  AS  int8)  INTO
idComprobanteItem;
     SELECT  CAST("Token"(texto,  delimitador,  3)  AS  int8)  INTO
idBitacora;
     SELECT  CAST("Token"(texto,  delimitador,  4)  AS  float8)  INTO
saldo;
     SELECT  CAST("Token"(texto,  delimitador,  5)  AS  float8)  INTO
abono;
     SELECT  CAST("Token"(texto,  delimitador,  6)  AS  varchar)  INTO
observacion;

       UPDATE
        "ComprobanteItem"
       SET
       "IdBitacora" = idBitacora,
       "Saldo" = saldo,
       "Abono" = abono,
       "Observacion" = observacion
       WHERE
       "IdComprobanteItem" = idComprobanteItem;
     ELSE
        EXIT;
     END IF;
  END LOOP;

 /* Actualización de los cobros. */
  campo:=0;
  LOOP
     campo:= campo + 1;
     delimitador:='#';
     SELECT "Token"(abonos, delimitador, campo) INTO texto;
     IF texto <> '' THEN
     delimitador:='|';
```

```
    SELECT  CAST("Token"(texto,  delimitador,  1)  AS  int8)  INTO
idAbono;
    SELECT  CAST("Token"(texto,  delimitador,  3)  AS  int8)  INTO
idCatalogoValor;

    SELECT  CAST("Token"(texto,  delimitador,  4)  AS  int8)  INTO
idBanco;
    SELECT  CAST("Token"(texto,  delimitador,  5)  AS  int8)  INTO
idTarjetaCredito;
    SELECT CAST("Token"(texto, delimitador, 6) AS varchar) INTO
cuentaCorriente;
    SELECT CAST("Token"(texto, delimitador, 7) AS varchar) INTO
secuenciaAbono;
    SELECT CAST("Token"(texto, delimitador, 8) AS varchar) INTO
fechaPlazo;
    SELECT  CAST("Token"(texto,  delimitador,  9)  AS  float8)  INTO
valorAbono;
    SELECT  CAST("Token"(texto,  delimitador,  10)  AS  int8)  INTO
idBitacoraAbono;
    SELECT  CAST("Token"(texto,  delimitador,  11)  AS  int8)  INTO
idEstadoAbono;

    UPDATE
         "Abono"
    SET
         "IdCatalogoValor" = idCatalogoValor,
         "IdBanco" = idBanco,
         "IdTarjetaCredito" = idTarjetaCredito,
         "CuentaCorriente" = cuentaCorriente,
         "Secuencia" = secuenciaAbono,
         "FechaPlazo" = fechaPlazo,
         "Valor" = valorAbono,
              "IdBitacora" = idBitacoraAbono,
              "IdEstado" = idEstadoAbono
    WHERE
         "IdAbono" = idAbono;
    ELSE
        EXIT;
     END IF;
  END LOOP;
  RETURN;
END;
$BODY$
LANGUAGE 'plpgsql' VOLATILE;
```

Función para eliminar un comprobante de la base:

```
CREATE  OR  REPLACE  FUNCTION  "eliminarComprobante"(idComprobamte
int8)
  RETURNS void AS
$BODY$
BEGIN
    DELETE FROM     "ComprobanteItem"  WHERE  "IdComprobante"  =
idComprobamte;
```

```
DELETE FROM    "Abono" WHERE "IdComprobante" = idComprobamte;
    DELETE FROM     "Comprobante" WHERE "Id" = idComprobamte;
    RETURN;
END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE;
```

4.1.7 Objetos de negocio

Clase que maneja la lógica de negocios del comprobante:

```
package com.cisc.erp.bo.tesoreria;

import com.cisc.erp.be.tesoreria.*;
import com.cisc.erp.dao.tesoreria.*;
import com.cisc.erp.exception.tesoreria.*;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;

public class ComprobanteBo {

    public   void   guardar(Comprobante   oComprobante)   throws
Exception, SQLException{
         /* Determinar insertar o modificar. */
         if (oComprobante.getIdComprobante() == 0) {
              insertar(oComprobante);
         } else {
              actualizar(oComprobante);
         }
    }

    private   void   insertar(Comprobante   oComprobante)   throws
Exception, SQLException{
         ArrayList documentosDescargados= null;
         validar(oComprobante);
         separarItem(oComprobante);
         documentosDescargados = descargar(oComprobante);
         Connection con = null;
         try {
              con = Conexion.getConnection();
              BitacoraDao      oBitacoraCobroDao      =      new
BitacoraDao(con);
              ComprobanteDao      oComprobanteDao      =      new
ComprobanteDao(con);
              oComprobanteDao.insertar(oComprobante);

              for (int i = 0; i < documentosDescargados.size();
i++) {

    oBitacoraCobroDao.actualizar((Bitacora)documentosDescargados
.get(i));
              }
              con.commit();
```

```
        } catch (Exception e) {
             if (con != null) {
                  con.rollback();
                  throw new TesoreriaException(e.getMessage());
             }
        } finally {
             if (con != null) {
                  con.close();
             }
        }
    }

    public Comprobante seleccionar(String IdComprobante) throws
Exception, SQLException{
        Comprobante oComprobante;
        Connection con = null;
        try {
             con = Conexion.getConnection();
             ComprobanteDao      oComprobanteDao      =      new
ComprobanteDao(con);
             oComprobante                                        =
oComprobanteDao.seleccionar(IdComprobante);
        } catch (Exception e) {
             throw new TesoreriaException(e.getMessage());
        } finally {
             if (con != null) {
                  con.close();
             }
        }
        return oComprobante;
    }

    private  void  actualizar(Comprobante  oComprobante)  throws
Exception, SQLException{
        validar(oComprobante);
        Connection con = null;
        try {
             con = Conexion.getConnection();
             ComprobanteDao      oComprobanteDao      =      new
ComprobanteDao(con);
             oComprobanteDao.actualizar(oComprobante);
             con.commit();
        } catch (Exception e) {
             if (con != null) {
                  con.rollback();
                  throw new TesoreriaException(e.getMessage());
                  }
        } finally {
             if (con != null) {
                  con.close();
             }
        }
    }

    public  void  eliminar(int  IdComprobante)  throws  Exception,
SQLException{
        Connection con = null;
```

```
        try {
             con = Conexion.getConnection();
             ComprobanteDao      oComprobanteDao      =      new
ComprobanteDao(con);
             oComprobanteDao.eliminar(IdComprobante);
             con.commit();} catch (Exception e) {
             if (con != null) {
                  con.rollback();
                  throw new TesoreriaException(e.getMessage());
             }
        } finally {
             if (con != null) {
                  con.close();
             }
        }
    }

    private   void   validar(Comprobante   oComprobante)   throws
Exception, NullPointerException {
        ArrayList oItems = null;
        BitacoraBo oBitacoraBo = null;
        CatalogoValorBo oCatalogoValorBo = null;

        /* Validación de cabecera comprobante. */
        if (oComprobante.getValor() == 0.0f) {
             throw       new       TesoreriaException("Cabecera
comprobante: Documento vacío...");
        }

        if (oComprobante.getIdTipoDocumento().equals("")) {
             throw       new       TesoreriaException("Cabecera
comprobante: Falta tipo documento...");
        }

        if (oComprobante.getIdSucursal() == 0) {
             throw       new       TesoreriaException("Cabecera
comprobante: Falta sucursal...");
        }

        if (oComprobante.getIdPersona() == 0) {
             throw       new       TesoreriaException("Cabecera
comprobante: Falta persona...");
        }

        if (oComprobante.getValorAbono() == 0.0f) {
             throw       new       TesoreriaException("Cabecera
comprobante: Falta valor del cobro...");
        }

        if            (oComprobante.getValor()             !=
oComprobante.getValorAbono()) {
             throw new TesoreriaException("Deuda cancelada debe
ser igual al valor abonado...");
        }

        if (oComprobante.getIdCajaTurno() == 0) {
             throw new TesoreriaException("Cabecera
```

```
comprobante: Falta de turno...");
         }

         /*  Validación  de  documentos  deudores  del  comprobante.
*/


 oItems = oComprobante.getItems();
        oBitacoraBo = new BitacoraBo();

         for (int i = 0; i < oItems.size(); i++) {
              ComprobanteItem         oComprobanteItem         =
(ComprobanteItem) oItems.get(i);
              Bitacora                oBitacora                =
oBitacoraBo.seleccionar(oComprobanteItem.getIdBitacora());
              /*  Valido  sólo  aquellos  documentos  que  han
abonado. */
              if (oComprobanteItem.getAbono() > 0) {
                   if (oBitacora.getCredito() == 2){
                        if       (oBitacora.getTotal()        !=
oComprobanteItem.getAbono()){
                             throw                         new
TesoreriaException("Deuda debe ser cancelada al contado...");
                        }
                   }
              }
         }

         oItems = oComprobante.getAbonos();
         oCatalogoValorBo = new CatalogoValorBo();

         for (int i = 0; i < oItems.size(); i++) {
              Abono oAbono = (Abono) oItems.get(i);
              CatalogoValor            oCatalogo             =
oCatalogoValorBo.seleccionar(oAbono.getIdCatalogoValor());

              if (oAbono.getValor() > 0.0f) {
                   if  ((oAbono.getIdCatalogoValor()  ==  1)  ||
(oAbono.getIdCatalogoValor()          ==          5)          ||
(oAbono.getIdCatalogoValor()          ==          6)          ||
(oAbono.getIdCatalogoValor()          ==          7)          ||
(oAbono.getIdCatalogoValor()          ==          8)          ||
(oAbono.getIdCatalogoValor() == 9)) {
                        if (oAbono.getIdBanco() != 0) {
                             throw                         new
TesoreriaException(oCatalogo.getNombre()  +  ":  No  especificar
banco.");
                        }
                   } else {
                        if (oAbono.getIdBanco() == 0) {
                             throw                         new
TesoreriaException(oCatalogo.getNombre()    +    ":    Especificar
banco.");
                        }
                   }
```

```
                    if  ((oAbono.getIdCatalogoValor()  ==  1)  ||
(oAbono.getIdCatalogoValor()          ==          2)           ||
(oAbono.getIdCatalogoValor()          ==          4)           ||
(oAbono.getIdCatalogoValor()          ==          5)           ||
(oAbono.getIdCatalogoValor()          ==          6)           ||
(oAbono.getIdCatalogoValor() == 7) ||

(oAbono.getIdCatalogoValor()          ==          8)           ||
(oAbono.getIdCatalogoValor() == 9)) {
                         if  (oAbono.getIdTarjetaCredito()  !=  0)
{
                              throw                            new
TesoreriaException(oCatalogo.getNombre()  +  ":  No  especificar
tarjeta de crédito.");
                         }
                    } else {
                         if  (oAbono.getIdTarjetaCredito()  ==  0)
{
                              throw                            new
TesoreriaException(oCatalogo.getNombre()   +   ":   Especificar
tarjeta de crédito.");
                         }
                    }

                    if  ((oAbono.getIdCatalogoValor()  ==  1)  ||
(oAbono.getIdCatalogoValor()          ==          5)           ||
(oAbono.getIdCatalogoValor()          ==          6)           ||
(oAbono.getIdCatalogoValor()          ==          7)           ||
(oAbono.getIdCatalogoValor()          ==          8)           ||
(oAbono.getIdCatalogoValor() == 9)) {
                         if
(!oAbono.getCuentaCorriente().equals("")) {
                              throw                            new
TesoreriaException(oCatalogo.getNombre()  +  ":  No  especificar
cuenta corriente.");
                         }
                    } else {
                         if
(oAbono.getCuentaCorriente().equals("")) {
                              throw                            new
TesoreriaException(oCatalogo.getNombre() + ": Especificar cuenta
corriente.");
                         }
                    }

                    if  ((oAbono.getIdCatalogoValor()  ==  1)  ||
(oAbono.getIdCatalogoValor() == 4)) {
                         if   (!oAbono.getSecuencia().equals(""))
{
                              throw                            new
TesoreriaException(oCatalogo.getNombre()  +  ":  No  especificar
secuencia.");
                         }
                    } else {
                         if (oAbono.getSecuencia().equals("")) {
                              throw                            new
TesoreriaException(oCatalogo.getNombre() + ": Especificar
```

```
secuencia.");
                         }
                    }
               }
          }
}

private   void   separarItem(Comprobante   oComprobante)   throws
Exception, NullPointerException {
          ArrayList IdBitacoras = new ArrayList();
          ArrayList oItems = oComprobante.getItems();

          for(int i = 0; i < oItems.size(); i++){
               ComprobanteItem              oItem               =
(ComprobanteItem)oItems.get(i);

               if (oItem.getAbono() <= 0) {
                    IdBitacoras.add(oItem);
               }
          }

          for(int i = 0; i < IdBitacoras.size(); i++){
               ComprobanteItem              oItem               =
(ComprobanteItem)IdBitacoras.get(i);
               oComprobante.quitarItem(oItem.getIdBitacora());
          }

          ArrayList identificadores = new ArrayList();
          oItems = oComprobante.getAbonos();

          for(int i = 0; i < oItems.size(); i++){
               Abono oItem = (Abono)oItems.get(i);

               if (oItem.getValor() <= 0) {
                    identificadores.add(oItem);
               }
          }

          for(int i = 0; i < identificadores.size(); i++){
               Abono oItem = (Abono)identificadores.get(i);

    oComprobante.quitarAbono(oItem.getIdentificador());
          }
    }

    private ArrayList descargar(Comprobante oComprobante) throws
Exception, NullPointerException{
          /* Disminuye el saldo del documento cancelado. */
          BitacoraBo oBitacoraBo = new BitacoraBo();
          ArrayList bitacoras = new ArrayList();
          ArrayList             documentosCancelados            =
oComprobante.getItems();

          for (int i = 0; i < documentosCancelados.size(); i++) {
               ComprobanteItem              oItem               =
(ComprobanteItem)documentosCancelados.get(i);
```

```
                Bitacora              oBitacoraCobro             =
oBitacoraBo.seleccionar(oItem.getIdBitacora());
                oBitacoraCobro.setSaldo(oBitacoraCobro.getSaldo()
- oItem.getAbono());
                bitacoras.add(oBitacoraCobro);
          }
          documentosCancelados = oComprobante.getAbonos();
          for (int i = 0; i < documentosCancelados.size(); i++) {
                Abono oItem = (Abono)documentosCancelados.get(i);

                Bitacora              oBitacoraCobro             =
oBitacoraBo.seleccionar(oItem.getIdBitacora());
                oBitacoraCobro.setSaldo(oBitacoraCobro.getSaldo()
- oItem.getValor());
                bitacoras.add(oBitacoraCobro);
          }
          return bitacoras;
    }

    private ArrayList cargar(Comprobante oComprobante) throws
Exception, NullPointerException{
          /* Aumenta el saldo del documento cancelado en caso de
que el comprobante sea anulados. */
          BitacoraBo oBitacoraBo = new BitacoraBo();
          ArrayList bitacoras = new ArrayList();
          ArrayList            documentosCancelados            =
oComprobante.getItems();

          for (int i = 0; i < documentosCancelados.size(); i++) {
                ComprobanteItem               oItem              =
(ComprobanteItem)documentosCancelados.get(i);
                Bitacora              oBitacoraCobro             =
oBitacoraBo.seleccionar(oItem.getIdBitacora());
                oBitacoraCobro.setSaldo(oBitacoraCobro.getSaldo()
+ oItem.getAbono());
                bitacoras.add(oBitacoraCobro);
          }
          documentosCancelados = oComprobante.getAbonos();
          for (int i = 0; i < documentosCancelados.size(); i++) {
                Abono oItem = (Abono)documentosCancelados.get(i);

                Bitacora              oBitacoraCobro             =
oBitacoraBo.seleccionar(oItem.getIdBitacora());
                oBitacoraCobro.setSaldo(oBitacoraCobro.getSaldo()
+ oItem.getValor());
                bitacoras.add(oBitacoraCobro);
          }
          return bitacoras;
    }

    public ArrayList busqueda(String tipoDocumento, String
secuencia, String fechaInicio, String fechaFin, String rucCi,
String apellidos, String nombres) throws Exception,
SQLException{
          /* Tratamiento de cadena. */
          tipoDocumento = tipoDocumento.trim().toUpperCase();
          secuencia = secuencia.trim().toUpperCase();
```

```
          fechaInicio = fechaInicio.trim().toUpperCase();
          fechaFin = fechaFin.trim().toUpperCase();
          rucCi = rucCi.trim().toUpperCase();
          apellidos = apellidos.trim().toUpperCase();
          nombres = nombres.trim().toUpperCase();
          rucCi = "%" + rucCi + "%";
          secuencia = "%" + secuencia + "%";
          apellidos = "%" + apellidos + "%";
          nombres = "%" + nombres + "%";

          ArrayList comprobantes;
          Connection con = null;
          try {
               con = Conexion.getConnection();
               ComprobanteDao      oComprobanteDao      =      new
ComprobanteDao(con);
               comprobantes                                       =
oComprobanteDao.busqueda(tipoDocumento,  secuencia,  fechaInicio,
fechaFin, rucCi, apellidos, nombres);
          } catch (Exception e) {
               throw new TesoreriaException(e.getMessage());
          } finally {
               if (con != null) {
                    con.close();
               }
          }
          return comprobantes;
    }
}
```

4.2 Vista

Comprobante.jsp contiene las páginas necesarias para la interfaz del comprobante:

```
<html>

<head>
<title>COMPROBANTE</title>
</head>

<body>

<div align="center">
  <center>
  <table  border="0"  cellspacing="1"  style="border-collapse:  collapse"
bordercolor="#111111" width="100%" id="AutoNumber1">
    <tr>
      <td         width="100%"><jsp:include         page="Cabecera.jsp"
flush="true"/></td>
    </tr>
    <tr>
```

```
      <td        width="100%"><jsp:include       page="MenuDefault.jsp"
flush="true"/></td>
    </tr>
    <tr>
      <td width="100%"> </td>
    </tr>
    <tr>
      <td    width="100%"><jsp:include    page="CabeceraComprobante.jsp"
flush="true"/> </td>
    </tr>
    <tr>
      <td      width="100%"><jsp:include      page="ItemComprobante.jsp"
flush="true"/> </td>
    </tr>
    <tr>
      <td         width="100%"><jsp:include         page="ItemAbono.jsp"
flush="true"/> </td>
    </tr>
    <tr>
      <td     width="100%"><jsp:include     page="OpcionComprobante.jsp"
flush="true"/> </td>
    </tr>
    <tr>
      <td            width="100%"><jsp:include            page="Pie.jsp"
flush="true"/> </td>
    </tr>
  </table>
  </center>
</div>

</body>

</html>
```

Cabecera.jsp es la página de encabezado para toda la aplicación:

```
<table border=0 width="100%" style="border-width: 0">
    <tr border=0 height=10>
    <td  border=0  width="23%"  style="border-style:  none;  border-width:
medium;
filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr='#00400
0',           endColorstr='#6B7F50',           gradientType='0');"><img
src="images\DINERO.JPG" width=237 height=60 /></td>
    <td border=0 width="77%" align="center" style="border-style: none;
border-width:                                                  medium;
filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr='#00400
0', endColorstr='#6B7F50', gradientType='0');"><font face="Courier New"
color="#FFFFFF"      size="6"><b>SISTEMA      DE      GESTIÓN      DE
TESORERÍA</b></font></td>
    </tr>
  </table>
```

CabeceraComprobante.jsp presenta el encabezado del comprobante:

```
<%@page import = "com.cisc.erp.be.tesoreria.Bitacora"%>
<%@page import = "com.cisc.erp.be.tesoreria.Comprobante"%>

<%



   Bitacora oPersona = (Bitacora)session.getAttribute("oPersona");
   Comprobante                      oComprobante                      =
(Comprobante)session.getAttribute("oComprobante");
%>
<html>
<head>
<title>COMPROBANTE</title>
   <script         language="JavaScript"         type="text/javascript"
src="script/Utilidad.js"></script>
</head>

<body
onLoad=frmCabeceraComprobante.txtCedula.focus();setIndiceFormulario()
bgcolor="#6B7F50">
<table    width="100%"    height="27"    border="1"    cellpadding="0"
cellspacing="1"  style="border-collapse:  collapse;  font-family:Courier
New;      font-size:10pt;      color:#000000"      bordercolor="#FFFFFF"
id="AutoNumber1">
  <tr>
    <form method="POST" action="ControladorComprobante.jsp">
    <td    width="100%"    height="18"    align="center"    colspan="8"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000', endColorstr='#6B7F50', gradientType='0');">
       <input                type="hidden"                name="pagina"
value="CabeceraComprobante.jsp">
       <input type="hidden" name="indiceForm" value="0">
       <input  type="hidden"  name="accionComprobante"  value="2"><!--
Consulta de un comprobante. -->
       <font                                                    size="3"
color="#FFFFFF"><%=oComprobante.getIdTipoDocumento()%></font><input
type="text"    name="txtIdComprobante"    size="15"    maxlength="15"
value="<%=oComprobante.getSecuencia()%>"  style="font-family:  Courier
New; font-size: 10pt; color: #000000"><input type="button" value="..."
name="btnBuscarComprobante"
onclick="ventanaSecundaria('BuscarComprobante.jsp?indiceForm='        +
indiceForm.value          +          '&tipoDocumento='                +
'<%=oComprobante.getIdTipoDocumento()%>',                   'ventana1',
'width=660,height=200,top=333,left=180,scrollbars=yes')">
    </td>
    </form>
  </tr>
  <tr>
    <td  width="50%"  style="border-right-style:  none;  border-right-
width:   medium;   border-bottom-style:   none;   border-bottom-width:
medium"><font                                                   size="3"
color="#FFFFFF">CIUDAD:</font><%=oComprobante.getIdSucursal()%> </
td>
    <td width="50%" style="border-left-style: none; border-left-width:
```

```
 medium; border-bottom-style: none; border-bottom-width: medium"><font size="3"
color="#FFFFFF">FECHA:</font><%=oComprobante.getFechaRegistro()%> 
</td>
  </tr>
  <tr>
    <form method="POST" name="frmCabeceraComprobante"
action="ControladorComprobante.jsp">
    <td width="50%" style="border-right-style: none; border-right-
width: medium; border-top-style: none; border-top-width: medium;
border-bottom-style: none; border-bottom-width: medium">
       <input type="hidden" name="pagina"
value="CabeceraComprobante.jsp">
       <input type="hidden" name="indiceForm" value="0">
       <input type="hidden" name="accionComprobante" value="10"><!--
Inicia búsqueda de documentos deudores y acreedores de una persona.  --
>
       <input type="hidden" name="tipoComprobante"
value="<%=oComprobante.getIdTipoDocumento()%>">
       <font size="3" color="#FFFFFF">RUC-CI:</font> <input
type="text" name="txtCedula" size="13" maxlength="13"
value="<%=oPersona.getRucCi()%>" style="font-family: Courier New; font-
size: 10pt; color: #000000"><input type="button" value="..."
name="btnBuscarPersona"
onclick="ventanaSecundaria('BuscarPersona.jsp?indiceForm='+indiceForm.v
alue, 'ventana1',
'width=660,height=200,top=333,left=180,scrollbars=yes')">
    </td>
    </form>
    <td width="50%" style="border-left-style: none; border-left-width:
medium; border-top-style: none; border-top-width: medium; border-
bottom-style: none; border-bottom-width: medium"><font size="3"
color="#FFFFFF">SR(S):</font><%=oPersona.getApellidos() + " " +
oPersona.getNombres()%> </td>
  </tr>
  <tr>
    <td width="50%" style="border-right-style: none; border-right-
width: medium; border-top-style: none; border-top-width: medium;
border-bottom-style: none; border-bottom-width: medium">
    <font size="3"
color="#FFFFFF">DIRECCIÓN:</font><%=oPersona.getDireccion()%> </td
>
    <td width="50%" style="border-left-style: none; border-left-width:
medium; border-top-style: none; border-top-width: medium; border-
bottom-style: none; border-bottom-width: medium"><font size="3"
color="#FFFFFF">TELÉFONO:</font><%=oPersona.getTelefono()%> </td>
  </tr>
  <tr>
    <td width="5E0%" style="border-right-style: none; border-right-
width: medium; border-top-style: none; border-top-width: medium">
    <font size="3"
color="#FFFFFF">CAJERO:</font><%=oComprobante.getIdCajaTurno()%> <
/td>
    <td width="50%" style="border-left-style: none; border-left-width:
medium; border-top-style: none; border-top-width: medium"><font
size="3"
color="#FFFFFF">TOTAL:</font><%=oComprobante.getValor()%> </td>
```

```
  </tr>
</table>
</body>
</html>
```

ItemCOmprobante.jsp presenta los items que pertenecen a un comprobante:

```
<%@page import = "com.cisc.erp.be.tesoreria.Comprobante"%>
<%@page import = "com.cisc.erp.be.tesoreria.ComprobanteItem"%>
<%@page import = "com.cisc.erp.be.tesoreria.Bitacora"%>
<%@page import = "com.cisc.erp.bo.tesoreria.BitacoraBo"%>
<%@page import = "java.util.ArrayList"%>

<html>

<head>
<title>DEUDA(S)</title>
</head>

<body bgcolor="#6B7F50">
<fieldset style="border:1px solid #FFFFFF; padding:2; ">
  <legend><font               color="#FFFFFF"              face="Courier
New">DETALLE:</font></legend>

<table    width="100%"    height="27"    border="1"    cellpadding="0"
cellspacing="1"  style="border-collapse:  collapse;  font-family:Courier
New;      font-size:10pt;      color:#000000"      bordercolor="#FFFFFF"
id="AutoNumber1">
    <tr>
      <td    width="100%"    height="18"    align="center"    colspan="8"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000', endColorstr='#6B7F50', gradientType='0');">
      <font color="#FFFFFF" size="3">DOCUMENTOS DEUDORES</font></td>
    </tr>
    <tr>
      <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',       endColorstr='#6B7F50',       gradientType='0');"><font
color="#FFFFFF" size="3">FECHA</font></td>
      <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',       endColorstr='#6B7F50',       gradientType='0');"><font
color="#FFFFFF" size="3">TIPO</font></td>
      <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',       endColorstr='#6B7F50',       gradientType='0');"><font
color="#FFFFFF" size="3">SECUENCIA</font></td>
      <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',       endColorstr='#6B7F50',       gradientType='0');"><font
color="#FFFFFF" size="3">OBSERVACIÓN</font></td>
      <td width="0%" height="18" align="center"
```

```
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',       endColorstr='#6B7F50',       gradientType='0');"><font
color="#FFFFFF" size="3">TOTAL</font></td>
      <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',       endColorstr='#6B7F50',       gradientType='0');"><font
color="#FFFFFF" size="3">SALDO</font></td>
      <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000', endColorstr='#6B7F50', gradientType='0');"><font
color="#FFFFFF" size="3">ABONO</font></td>
      <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',       endColorstr='#6B7F50',       gradientType='0');"><font
color="#FFFFFF" size="3">ENVÍO</font></td>
    </tr>
<%
   float totalDeuda = 0.0f;
   float totalSaldo = 0.0f;
   ArrayList oItems = null;

   BitacoraBo                       oBitacoraBo                        =
(BitacoraBo)session.getAttribute("oBitacoraBo");
   Comprobante                      oComprobante                       =
(Comprobante)session.getAttribute("oComprobante");
   Bitacora oBitacora;
   oItems = oComprobante.getItems();

   for(int i=0; i<oItems.size(); i++){
      ComprobanteItem oItem = (ComprobanteItem)oItems.get(i);
      oBitacora = oBitacoraBo.seleccionar(oItem.getIdBitacora());
      totalDeuda = totalDeuda + oItem.getSaldo();
%>
    <tr>
    <form method="POST" action="ControladorComprobante.jsp">
      <input type="hidden" name="pagina" value="ItemComprobante.jsp">
      <input type="hidden" name="indiceForm" value="0">
      <input type="hidden" name="accionComprobante" value="5">
      <input           type="hidden"           name="idComprobanteItem"
value="<%=oItem.getIdComprobanteItem()%>">
      <input             type="hidden"             name="idComprobante"
value="<%=oItem.getIdComprobante()%>">
      <input               type="hidden"               name="idBitacora"
value="<%=oItem.getIdBitacora()%>">
      <td                    width="0%"                     height="18"
align="center"><%=oBitacora.getFechaDocumento()%> </td>
      <td                    width="0%"                     height="18"
align="center"><%=oBitacora.getIdTipoDocumento()%> </td>
      <td                    width="0%"                     height="18"
align="center"><%=oBitacora.getSecuencia()%> </td>
      <td width="0%" height="18" align="center">
      <input  name="txtObservacion"  value="<%=oItem.getObservacion()%>"
size="49"  maxlength="100"  style="font-family:  Courier  New;  text-
align:left"></td>
      <td                    width="0%"                     height="18"
align="right"><%=oBitacora.getTotal()%> </td>
```

```
      <td  width="0%"  height="18"  align="right"><%=oItem.getSaldo()  -
oItem.getAbono()%> </td>
      <td width="0%" height="18" align="center"><input name="txtAbono"
value="<%=oItem.getAbono()%>"  size="10"  maxlength="10"  style="font-
family:  Courier  New;  font-size:  10pt;  color:  #000000;  text-align:
right"
      onKeypress="
                if (event.keyCode < 46 || event.keyCode > 57){
                   event.returnValue = false;
                }
                "
      onblur="
                var saldo = Math.abs('<%=oItem.getSaldo()%>');
                var abono = Math.abs(txtAbono.value);

                if (isNaN(abono)) {
                 abono= 0.0;
                 }

                    txtAbono.value = abono;

                if (saldo < abono){
                   window.alert('Abono $' + abono + ' no debe ser mayor
que $' + saldo);
                   txtAbono.focus();
                }
      "></td>
      <td  width="0%"  height="18"  align="center"><input  type="submit"
value="Enviar" name="btnEnviar" style="font-family: Courier New; font-
size: 10pt"></td>
    </form>
    </tr>
<%}
   totalSaldo = totalDeuda - oComprobante.getValor();
%>
    <tr>
      <td  width="0%"  height="18"  style="border-left-width:  medium;
border-bottom-style:      none;      border-bottom-width:      medium"
colspan="4"> </td>
      <td                    width="0%"                    height="18"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',       endColorstr='#6B7F50',       gradientType='0');"><font
color="#FFFFFF" size="3">SUMA:</font></td>
      <td                    width="0%"                    height="18"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',          endColorstr='#6B7F50',          gradientType='0');"
align="right"><font                                    color="#FFFFFF"
size="3"><%=totalSaldo%></font> </td>
      <td                    width="0%"                    height="18"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',          endColorstr='#6B7F50',          gradientType='0');"
align="right"><font                                    color="#FFFFFF"
size="3"><%=oComprobante.getValor()%></font> </td>
      <td                    width="0%"                    height="18"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000', endColorstr='#6B7F50', gradientType='0');"></td>
    </tr>
```

```
  </table>
  </font>
</fieldset>

</body>
</html>
```

ItemAbono.jsp presenta las formas de pago de un comprobante:

```
<html>
<head><title>Abonos: </title>
    <script         language="JavaScript"         type="text/javascript"
src="script/date-picker.js"></script>
    <script         language="JavaScript"         type="text/javascript"
src="script/Utilidad.js"></script>
</head>
<%@page import = "java.util.ArrayList"%>
<%@page import = "com.cisc.erp.be.tesoreria.Comprobante"%>
<%@page import = "com.cisc.erp.be.tesoreria.Abono"%>
<%@page import = "com.cisc.erp.be.tesoreria.CatalogoValor"%>
<%@page import = "com.cisc.erp.be.tesoreria.Banco"%>
<%@page import = "com.cisc.erp.be.tesoreria.TarjetaCredito"%>

<%
   Comprobante oComprobante = null;
   ArrayList abonos = null;
   ArrayList valores = null;
   ArrayList bancos = null;
   ArrayList tarjetas = null;

   oComprobante = (Comprobante)session.getAttribute("oComprobante");
   abonos = oComprobante.getAbonos();
   valores = (ArrayList)session.getAttribute("valores");
   bancos = (ArrayList)session.getAttribute("bancos");
   tarjetas = (ArrayList)session.getAttribute("tarjetas");
%>
<body bgcolor="#6B7F50">
<fieldset style="border:1px solid #FFFFFF; padding:2; ">
  <legend><font              color="#FFFFFF"              face="Courier
New">DETALLE:</font></legend>

<table    width="100%"    height="27"    border="1"    cellpadding="0"
cellspacing="1"  style="border-collapse:  collapse;  font-family:Courier
New;      font-size:10pt;      color:#000000"      bordercolor="#FFFFFF"
id="AutoNumber1">
  <tr>
    <td    width="100%"    height="18"    align="center"    colspan="8"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000', endColorstr='#6B7F50', gradientType='0');">
    <font size="3" color="#FFFFFF">ABONO</font></td>
  </tr>
  <tr>
    <td          width="0%"          height="18"          align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000', endColorstr='#6B7F50', gradientType='0');"><font size="3"
```

```
color="#FFFFFF">FORMA</font></td>
    <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',  endColorstr='#6B7F50',  gradientType='0');"><font  size="3"
color="#FFFFFF">BANCO</font></td>
    <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',  endColorstr='#6B7F50',  gradientType='0');"><font  size="3"
color="#FFFFFF">TARJETA</font></td>
    <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',  endColorstr='#6B7F50',  gradientType='0');"><font  size="3"
color="#FFFFFF">CUENTA</font></td>
    <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',  endColorstr='#6B7F50',  gradientType='0');"><font  size="3"
color="#FFFFFF">SECUENCIA</font></td>
    <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',  endColorstr='#6B7F50',  gradientType='0');"><font  size="3"
color="#FFFFFF">FECHA</font></td>
    <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',  endColorstr='#6B7F50',  gradientType='0');"><font  size="3"
color="#FFFFFF">VALOR</font></td>
    <td         width="0%"         height="18"         align="center"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',  endColorstr='#6B7F50',  gradientType='0');"><font  size="3"
color="#FFFFFF">ENVÍO</font></td>
  </tr>
<%
   for(int i=0; i < abonos.size(); i++){
      Abono oAbono = (Abono)abonos.get(i);
%>
  <tr>
    <form                 name="frmAbono"                 method="POST"
action="ControladorComprobante.jsp">
    <input type="hidden" name="indiceForm" value="0">
    <input type="hidden" name="accionComprobante" value="12">
    <input                 type="hidden"                 name="idAbono"
value="<%=oAbono.getIdAbono()%>">
    <input              type="hidden"              name="idComprobante"
value="<%=oAbono.getIdComprobante()%>">
    <input              type="hidden"              name="identificador"
value="<%=oAbono.getIdentificador()%>">
    <input             type="hidden"             name="idCatalogoValor"
value="<%=oAbono.getIdCatalogoValor()%>">
    <input                 type="hidden"                 name="idBanco"
value="<%=oAbono.getIdBanco()%>">
    <input            type="hidden"            name="idTarjetaCredito"
value="<%=oAbono.getIdTarjetaCredito()%>">
    <td width="0%" height="1" align="center">
    <select                                            name=cboValores
onchange="cboBancos.selectedIndex=0;cboTarjetas.selectedIndex=0"
style="font-family: Courier New; font-size: 10pt; color: #000000">
    <%for (int a = 0; a < valores.size(); a++) {
```

```
        CatalogoValor oValor = (CatalogoValor)valores.get(a);
        if               (oAbono.getIdCatalogoValor()            ==
oValor.getIdCatalogoValor()) {
    %>
        <option              value="<%=oValor.getIdCatalogoValor()%>"
selected><%=oValor.getNombre()%></option>
    <%} else {%>
        <option
value="<%=oValor.getIdCatalogoValor()%>"><%=oValor.getNombre()%></optio
n>
    <%}}%>
    </select></td>

    <td width="0%" height="1" align="center">
    <select name=cboBancos style="font-family: Courier New; font-size:
10pt; color: #000000">
    <%for (int b = 0; b < bancos.size(); b++) {
        Banco oBanco = (Banco)bancos.get(b);
        if (oAbono.getIdBanco() == oBanco.getIdBanco()) {
    %>
        <option                      value="<%=oBanco.getIdBanco()%>"
selected><%=oBanco.getNombre()%></option>
    <%} else {%>
        <option
value="<%=oBanco.getIdBanco()%>"><%=oBanco.getNombre()%></option>
    <%}}%>
    </select></td>
    <td width="0%" height="1" align="center">
    <select  name=cboTarjetas  style="font-family:  Courier  New;  font-
size: 10pt; color: #000000">
     <%for (int c = 0; c < tarjetas.size(); c++) {
        TarjetaCredito               oTarjetaCredito                 =
(TarjetaCredito)tarjetas.get(c);
        if              (oAbono.getIdTarjetaCredito()                ==
oTarjetaCredito.getIdTarjetaCredito()) {
    %>
        <option     value="<%=oTarjetaCredito.getIdTarjetaCredito()%>"
selected><%=oTarjetaCredito.getNombre()%></option>
    <%} else {%>
        <option
value="<%=oTarjetaCredito.getIdTarjetaCredito()%>"><%=oTarjetaCredito.g
etNombre()%></option>
    <%}}%>
    </select></td>

    <td   width="0%"   height="1"   align="center"><input   type="text"
name="txtCuenta"                size="15"                maxlength="15"
value="<%=oAbono.getCuentaCorriente()%>"  style="font-family:  Courier
New; font-size: 10pt; color: #000000"
        onKeypress="
               if  ((event.keyCode  >=  48  &&  event.keyCode  <=  57)  ||
(event.keyCode  ==  45)  ||  (event.keyCode  ==  32)  ||  (event.keyCode  ==
255)){
                  event.returnValue = true;
               } else {
                  event.returnValue = false;
               }
```

```
                ">
    </td>
    <td   width="0%"   height="1"   align="center"><input   type="text"
name="txtSecuencia"              size="15"              maxlength="15"
value="<%=oAbono.getSecuencia()%>"  style="font-family:  Courier  New;
font-size: 10pt; color: #000000"
            onKeypress="
                if  ((event.keyCode  >=  48  &&  event.keyCode  <=  57)  ||
(event.keyCode  ==  45)  ||  (event.keyCode  ==  32)  ||  (event.keyCode  ==
255)){
                   event.returnValue = true;
                } else {  event.returnValue = false;
                }
                ">
    </td>
    <td width="0%" height="1" align="center">
    <input  type="text"  name="txtFechaPlazo"  size="10"  maxlength="10"
value="<%=oAbono.getFechaPlazo()%>"  style="font-family:  Courier  New;
font-size:  10pt;  color:  #000000;  text-align:center"  onfocus="blur()"
onclick=calendario(indiceForm.value,'txtFechaPlazo')></td>
    <td width="0%" align="center" height="1">
    <input       name="txtValor"       size="10"       maxlength="10"
value="<%=oAbono.getValor()%>"  style="font-family:  Courier  New;  font-
size: 10pt; color: #000000; text-align:right"
    onKeypress="
                if (event.keyCode < 46 || event.keyCode > 57){
                   event.returnValue = false;
                }
                "
    onblur="
                var valor = Math.abs(txtValor.value);

                if (isNaN(valor)) {
                 txtValor.value = 0.0;
                }
    "></td>
    <td width="0%" height="1" nowrap align="center">
    <input  type="submit"  value="Enviar"  name="btnEnviar"  style="font-
family: Courier New; font-size: 10pt; color: #000000"></td>
    </form>
  </tr>
<%}%>
  <tr>
    <td height="18" colspan="5"> </td>
    <td                      width="0%"                      height="18"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',        endColorstr='#6B7F50',        gradientType='0');"><p
align="center"><font size="3" color="#FFFFFF">SUMA:</font></td>
    <td                      width="0%"                      height="18"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',        endColorstr='#6B7F50',        gradientType='0');"><p
align="right"><font                                          size="3"
color="#FFFFFF"><%=oComprobante.getValorAbono()%></font></td>
    <td                      width="0%"                      height="18"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000',         endColorstr='#6B7F50',         gradientType='0');"
align="left"></td>
```

```
  </tr>
</table>
</fieldset>
</body>
</html>
```

OpcionComprobante.jsp contiene las opciones que van a actuar sobre el comprobante:

```
<html>

<head>
<title>OPCIÓN</title>
</head>

<body bgcolor="#6B7F50">
<div align="center">
  <center>
<fieldset style="border:1px solid #FFFFFF; padding:2; ">
  <legend><font              color="#FFFFFF"              face="Courier
New">OPCIONES:</font></legend>
  <table  border="0"  cellspacing="1"  style="border-collapse:  collapse;
font-family: Courier New; font-size: 10pt; color: #000000" width="100%"
id="AutoNumber1">
    <tr>
     <form method="POST" action="ControladorComprobante.jsp">
      <td width="100%">
        <input type="hidden" name="pagina" value="Comprobante.jsp">
        <input type="hidden" name="indiceForm" value="0">
        <input type="hidden" name="accionComprobante" value="0">
        <p align="center">
        <input
onclick="forms[indiceForm.value].accionComprobante.value=1;
forms[indiceForm.value].willSubmit=confirm('¿Desea      guardar      el
documento?');return  forms[indiceForm.value].willSubmit;"  type="submit"
value="Guardar"   name="btnGuardar"   style="font-family:  Courier  New;
font-size: 10pt; color: #000000; text-align: center"> </td>
     </form>
    </tr>
  </table>
</fieldset>
  </center>
</div>
</body>

</html>
```

Pie.jsp es la página de pie para toda la aplicación:

```
<table border="0" cellpadding="0" cellspacing="0" style="border-
```

```
collapse:      collapse"      bordercolor="#111111"      width="100%"
id="AutoNumber1">
  <tr>
    <td                    width="50%"                    height="80%"
style="filter:progid:DXImageTransform.Microsoft.Gradient(startColorstr=
'#004000', endColorstr='#6B7F50', gradientType='0');" align="center">
    <b><font color="#FFFFFF" face="Courier New" size="2">CopyRights(c),
2006 Universidad Estatal de Guayaquil</font>
    </b>
    </td>
  </tr>
</table>
```

4.3 Controlador

ControladorConprobante.jsp contiene todas las acciones a ejecutarse en el comprobante:

```
<%@page import = "com.cisc.erp.be.tesoreria.Comprobante"%>
<%@page import = "com.cisc.erp.be.tesoreria.ComprobanteItem"%>
<%@page import = "com.cisc.erp.be.tesoreria.Abono"%>
<%@page import = "com.cisc.erp.be.tesoreria.Bitacora"%>
<%@page import = "com.cisc.erp.bo.tesoreria.ComprobanteBo"%>
<%@page import = "com.cisc.erp.bo.tesoreria.BitacoraBo"%>
<%@page import = "com.cisc.erp.bo.tesoreria.CatalogoValorBo"%>
<%@page import = "com.cisc.erp.bo.tesoreria.BancoBo"%>
<%@page import = "com.cisc.erp.bo.tesoreria.TarjetaCreditoBo"%>
<%@page import = "com.cisc.erp.exception.tesoreria.*"%>
<%@page import = "java.util.ArrayList"%>

<%
   String url = null;
   int accionComprobante;
   String idComprobante = null;
   String tipoComprobante = null;
   String cedula = null;
   int idSucursal; //¿Cómo lo obtendré?
   int idCajaTurno; //¿Cómo lo obtendré?
   int deudor;
   int acreedor;
   String estadoHabilitado = null;
   String estadoDesabilitado = null;
   String pagina = null;

   Comprobante oComprobante = null;
   Bitacora oPersona = null;
   ComprobanteBo oComprobanteBo = null;
   BitacoraBo oBitacoraBo = null;
   CatalogoValorBo oCatalogoValorBo = null;
   BancoBo oBancoBo = null;
   TarjetaCreditoBo oTarjetaCreditoBo = null;
```

```
   ArrayList valores = null;
   ArrayList bancos = null;
   ArrayList tarjetas = null;
   ArrayList documentosDeudores = null;
   ArrayList documentosAcreedores = null;

   estadoHabilitado = "1";
   estadoDesabilitado = "2";
   ComprobanteItem oComprobanteItem = null;
   Abono oAbono = null;

   accionComprobante = new
Integer(request.getParameter("accionComprobante")).intValue();

   switch (accionComprobante) {
      case 0:
         /* INICIAR. */
         /* Inicializa los objectos que colabararán con el comprobante:
oComprobante, oPersona. */

         tipoComprobante                                              =
(String)request.getParameter("tipoComprobante");
         url = "Comprobante.jsp";

         oComprobante = new Comprobante();
         oComprobante.setIdTipoDocumento(tipoComprobante);
         oPersona = new Bitacora();

         session.setAttribute("oComprobante", oComprobante);
         session.setAttribute("oPersona", oPersona);
         break;

      case 10:
         /*
            Elaborar un comprobante preliminar.
            Se tomará número de cédula para la búsqueda de documentos
deudores y acreedores.
            El comprobante se encuentra en modo de inserción.
         */

         cedula = (String)request.getParameter("txtCedula");
         oComprobante                                                 =
(Comprobante)session.getAttribute("oComprobante");
         tipoComprobante = oComprobante.getIdTipoDocumento();
         url = "Comprobante.jsp";

         if (cedula.equals("")) {
            url                                                       =
"ControladorComprobante.jsp?accionComprobante=0&tipoComprobante="     +
tipoComprobante;
         } else {
            oBitacoraBo = new BitacoraBo();
            oCatalogoValorBo = new CatalogoValorBo();
            oBancoBo = new BancoBo();
            oTarjetaCreditoBo = new TarjetaCreditoBo();

                idSucursal = 1; //¿Cómo lo obtendré?
```

```
            idCajaTurno = 3; //¿Cómo lo obtendré?
          deudor = 1;
           acreedor = 2;

            String idFactura = "FACT";
            int CatalogoFactura = 5;
            String idNotaVenta = "NV";
            int CatalogoNotaVenta = 6;
            String idComprobanteRetencion = "CR";
            int CatalogoComprobanteRetencion = 7;
            String idNotaCredito = "NC";
            int CatalogoNotaCredito = 8;
            String idNotaDebito = "ND";
            int CatalogoNotaDebito = 9;

            oPersona = oBitacoraBo.seleccionarPersona(cedula);
            /*   CI:   documentos   deudores   en   primer   detalle   del
comprobante.
                   documentos   acreedores   en   segundo   detalle   del
comprobante.

              OPG:   documentos   deudores   en   segundo   detalle   del
comprobante.
                   documentos   acreedores   en   primer   detalle   del
comprobante.
            */

            if (tipoComprobante.equals("CI")) {
               documentosDeudores    =    oBitacoraBo.documentos(cedula,
deudor);
               documentosAcreedores   =   oBitacoraBo.documentos(cedula,
acreedor);
            } else {
               documentosDeudores    =    oBitacoraBo.documentos(cedula,
acreedor);
               documentosAcreedores   =   oBitacoraBo.documentos(cedula,
deudor);
            }

            oComprobante = new Comprobante();
            oComprobante.setIdTipoDocumento(tipoComprobante);
            oComprobante.setIdSucursal(idSucursal);
            oComprobante.setIdPersona(oPersona.getIdPersona());
            oComprobante.setIdCajaTurno(idCajaTurno);

            /* Regla: Comprobante en modo de consulta traerá "TODOS"
los valores, bancos, tarjetas;
               caso contrario sólo los habilitados.
           */
           valores       =       oCatalogoValorBo.buscar("%",       "%",
estadoHabilitado);
            bancos = oBancoBo.buscar("%", "%", estadoHabilitado);
            tarjetas    =    oTarjetaCreditoBo.buscar("%",    "%",    "%",
estadoHabilitado);

            for (int i = 0; i < documentosDeudores.size(); i++) {
               oComprobanteItem = new ComprobanteItem();
```

```
               Bitacora                   oBitacora                   =
(Bitacora)documentosDeudores.get(i);

oComprobanteItem.setIdBitacora(oBitacora.getIdBitacora());
               oComprobanteItem.setSaldo(oBitacora.getSaldo());

               if (oBitacora.getCredito() == 1) {
                  oComprobanteItem.setObservacion("A crédito");
               } else {
                  oComprobanteItem.setObservacion("Al contado");
               }
               oComprobante.agregarItem(oComprobanteItem);
            }

            int i = 0;
            for (i = 0; i < documentosAcreedores.size(); i++) {

               oAbono = new Abono();
               Bitacora                   oBitacora                   =
(Bitacora)documentosAcreedores.get(i);

               if (idFactura.equals(oBitacora.getIdTipoDocumento())) {
                  oAbono.setIdentificador(i);
                  oAbono.setIdBitacora(oBitacora.getIdBitacora());
                  oAbono.setIdCatalogoValor(CatalogoFactura);
                  oAbono.setSecuencia(oBitacora.getSecuencia());
                  oAbono.setValor(oBitacora.getSaldo());
                  oComprobante.agregarAbono(oAbono);
               }
               if   (idNotaVenta.equals(oBitacora.getIdTipoDocumento()))
{
                  oAbono.setIdentificador(i);
                  oAbono.setIdBitacora(oBitacora.getIdBitacora());
                  oAbono.setIdCatalogoValor(CatalogoNotaVenta);
                  oAbono.setSecuencia(oBitacora.getSecuencia());
                  oAbono.setValor(oBitacora.getSaldo());
                  oComprobante.agregarAbono(oAbono);
               }
               if
(idComprobanteRetencion.equals(oBitacora.getIdTipoDocumento())) {
                  oAbono.setIdentificador(i);
                  oAbono.setIdBitacora(oBitacora.getIdBitacora());

oAbono.setIdCatalogoValor(CatalogoComprobanteRetencion);
                  oAbono.setSecuencia(oBitacora.getSecuencia());
                  oAbono.setValor(oBitacora.getSaldo());
                  oComprobante.agregarAbono(oAbono);
               }
               if
(idNotaCredito.equals(oBitacora.getIdTipoDocumento())) {
                  oAbono.setIdentificador(i);
                  oAbono.setIdBitacora(oBitacora.getIdBitacora());
                  oAbono.setIdCatalogoValor(CatalogoNotaCredito);
                  oAbono.setSecuencia(oBitacora.getSecuencia());
                  oAbono.setValor(oBitacora.getSaldo());
                  oComprobante.agregarAbono(oAbono);
               }
```

```
               if (idNotaDebito.equals(oBitacora.getIdTipoDocumento()))
{
                  oAbono.setIdentificador(i);
                  oAbono.setIdBitacora(oBitacora.getIdBitacora());
                  oAbono.setIdCatalogoValor(CatalogoNotaDebito);
                  oAbono.setSecuencia(oBitacora.getSecuencia());
                  oAbono.setValor(oBitacora.getSaldo());
                  oComprobante.agregarAbono(oAbono);
               }
            }

            if (documentosDeudores.size() > 0 ) {
               oAbono = new Abono();
               oAbono.setIdentificador(i);
               oComprobante.agregarAbono(oAbono);
            }

            session.setAttribute("oComprobante", oComprobante);
            session.setAttribute("oPersona", oPersona);
            session.setAttribute("oBitacoraBo", oBitacoraBo);
            session.setAttribute("valores", valores);
            session.setAttribute("bancos", bancos);
            session.setAttribute("tarjetas", tarjetas);
         }
         break;

       case 5:
         /* AFECTAR DOCUMENTOS DEUDORES. */
         /* Agregar los últimos cambios efectuados a los documentos
deudores del comprobante.*/

         url = "Comprobante.jsp";
         int              idBitacora              =              new
Integer(request.getParameter("idBitacora")).intValue();
         float                abono               =              new
Float(request.getParameter("txtAbono")).floatValue();
         String observacion = request.getParameter("txtObservacion");

         oComprobanteItem = new ComprobanteItem();
         oComprobanteItem.setIdBitacora(idBitacora);
         oComprobanteItem.setAbono(abono);
         oComprobanteItem.setObservacion(observacion);

         oComprobante                                                =
(Comprobante)session.getAttribute("oComprobante");
         oComprobante.agregarItem(oComprobanteItem);
         session.setAttribute("oComprobante", oComprobante);
            break;

       case 12:
         /* AGREGAR DOCUMENTOS ACREEDORES.*/

         url = "Comprobante.jsp";
         int              identificador             =            new
Integer(request.getParameter("identificador")).intValue();
         int             idCatalogoValor2            =           new
Integer(request.getParameter("cboValores")).intValue();
```

```
        int              idBanco              =             new
Integer(request.getParameter("cboBancos")).intValue();
        int           idTarjetaCredito           =          new
Integer(request.getParameter("cboTarjetas")).intValue();
        String cuentaCorriente = request.getParameter("txtCuenta");
        String secuencia = request.getParameter("txtSecuencia");
        String fechaPlazo = request.getParameter("txtFechaPlazo");
        float              valor              =             new
Float(request.getParameter("txtValor")).floatValue();

        oAbono = new Abono();
        oAbono.setIdentificador(identificador);
        oAbono.setIdCatalogoValor(idCatalogoValor2);
        oAbono.setIdBanco(idBanco);
        oAbono.setIdTarjetaCredito(idTarjetaCredito);
        oAbono.setCuentaCorriente(cuentaCorriente);
        oAbono.setSecuencia(secuencia);
        oAbono.setFechaPlazo(fechaPlazo);
        oAbono.setValor(valor);

        oComprobante                                            =
(Comprobante)session.getAttribute("oComprobante");
        oComprobante.agregarAbono(oAbono);
        oComprobante.agregarAbonoFinal(oAbono);
        session.setAttribute("oComprobante", oComprobante);
          break;

      case 1:
        /* INSERTAR Y ACTUALIZAR */
        /*  Ingresa  o  actualiza  información  de  comprobante:
oComprobante. Aumenta o disminuye saldo de bitacora. */

        try {
          oComprobanteBo = new ComprobanteBo();
          oComprobante                                          =
(Comprobante)session.getAttribute("oComprobante");
          tipoComprobante = oComprobante.getIdTipoDocumento();
          oComprobanteBo.guardar(oComprobante);

          url                                                   =
"ControladorComprobante.jsp?accionComprobante=0&tipoComprobante="     +
tipoComprobante;
        }catch(TesoreriaException e){
          url = "Exception.jsp?excepcion=" + e.toString();
        }
        break;

      case 2:
        /* SELECCIONAR */
        /* Trae información según número comprobante: oComprobante. */

        oCatalogoValorBo = new CatalogoValorBo();
        oBancoBo = new BancoBo();
        oTarjetaCreditoBo = new TarjetaCreditoBo();

        idComprobante = request.getParameter("txtIdComprobante");
        url = "Comprobante.jsp";
```

```
         oComprobanteBo = new ComprobanteBo();
         oComprobante = oComprobanteBo.seleccionar(idComprobante);
         oBitacoraBo = new BitacoraBo();
         oPersona                                                     =
oBitacoraBo.seleccionarPersona(oComprobante.getIdPersona());

           /*
            Regla:  Cuando  el  comprobante  se  encuentra  en  modo  de
consulta debe traer valores, bancos, tarjetas habilitados
            y desabilitados, caso contrario sólo los habilitados.
           */
         valores = oCatalogoValorBo.buscar("%", "%", "%");
         bancos = oBancoBo.buscar("%", "%", "%");
         tarjetas = oTarjetaCreditoBo.buscar("%", "%", "%", "%");

         session.setAttribute("oComprobante", oComprobante);
         session.setAttribute("oBitacoraBo", oBitacoraBo);
         session.setAttribute("oPersona", oPersona);
         session.setAttribute("valores", valores);
         session.setAttribute("bancos", bancos);
         session.setAttribute("tarjetas", tarjetas);
         break;



/***********************************************************************
*/
/* Ver si eliminar. */
      case 3:
         /* ANULAR */

         try {
            pagina = (String)request.getParameter("pagina");
            idComprobante                                             =
(String)request.getParameter("txtIdComprobante");

            oComprobanteBo = new ComprobanteBo();
            oComprobante = oComprobanteBo.seleccionar(idComprobante);
            oComprobante.setIdEstado(2);

            oComprobanteBo.guardar(oComprobante); //Aplicar  la  carga  o
descarga en bitacora.
            url = pagina;
         }catch(TesoreriaException e){
            url = "Exception.jsp?excepcion=" + e.toString();
         }
            break;

       case 4:
         /* ELIMINAR */

         oComprobanteBo = new ComprobanteBo();
         oComprobante                                                 =
(Comprobante)session.getAttribute("oComprobante");
         tipoComprobante = oComprobante.getIdTipoDocumento();
```

```
         oComprobanteBo.eliminar(oComprobante.getIdComprobante());
         url                                                        =
"ControladorComprobante.jsp?accionComprobante=0&tipoComprobante="     +
tipoComprobante;
            break;

       case 6:
         /* QUITAR */
         url = "QuitarComprobanteItem.jsp";
            break;

       case 7:
         /* MANEJAR ERROR */
         url = "ErrorComprobante.jsp";
            break;
      case 8:
         /*Objetos  a  null,  garantizando  el  siguiente  ingresa  de
comprobante.*/

         url = "ControladorComprobante.jsp?accionComprobante=0";
         oComprobanteBo = null;
         oBitacoraBo = null;
         oPersona = null;
         oComprobante = null;
         break;

      case 9:
         /* PRESENTAR */
         /*Muestra  contenido  de  objectos  que  colabarán  con  el
comprobante.*/

         url = "MostrarComprobante.jsp";
         break;

      case 11:
         /* PRESENTAR */
         /*Muestra  contenido  de  objectos  que  colabarán  con  el
comprobante.*/

         oComprobanteBo = new ComprobanteBo();

//oComprobanteBo.filtrar((Comprobante)session.getAttribute("oComprobant
e"));
         url = "ComprobanteIngreso.jsp";
         break;

      default:
         url = "Fecha.jsp";
         break;
   }
%>
<jsp:forward page="<%=url%>"/>
```

CAPÍTULO 5

5 PRUEBAS

5.1 El ciclo de vida FLOOT

De la amplia variedad de técnicas disponibles en todos los aspectos del desarrollo de software, hemos realizado las siguientes pruebas:

5.1.1 Prueba de caja negra

Verificamos que los ítem que se esten probando, cuando se dan las entradas apropiadas, produzcan los resultados esperados.

5.1.2 Prueba de clases

Nos aseguramos que las clases y todas sus instancias cumplan con el comportamiento definido .

5.1.3 Prueba de integración de clases

Nos asegurarmos que las clases, y sus instancias, conforman un software que cumpla con el comportamiento definido.

5.1.4 Prueba de componente

Validamos que los componentes funcionan tal como esta definido.

5.1.5 Prueba de integración

Realizarmos pruebas para verificar que las partes del software funcionan juntas.

5.1.6 Demostrar con el código

La mejor forma de determinar si un modelo realmente refleja lo que se necesita, o lo que se debe construir, es construyendo software basado en el modelo para mostrar que el modelo esta bien.

5.1.7 Prueba de regresión

Después de realizar algún cambio en la aplicación, nos aseguramos que los comportamientos previamente probados todavía trabajan como se espera.

5.1.8 Prueba de stress

Nos asegurarmos que el sistema funcione como se espera bajo grandes volúmenes de transacciones, usuarios, carga y demás.

5.1.9 Revisión técnica

Intercambiamos, entre los integrantesa del grupo, el diseño dpara que sea revisado de forma exhaustiva. Una revisión típicamente se enfoca en la precisión, calidad, facilidad de uso y completitud.

5.1.10 Prueba de interfaz de ususario

Probamos la interfaz de usuario para garantizar que cumple los estándares y requerimientos definidos.

CONCLUSIONES

A lo largo del desarrollo del Sistema de Gestión de Tesorería hemos adquirido conocimientos muy valiosos acerca de las herramientas Open Source. Entre estos conocimientos se incluyen:

Imporancia del ciclo de análisis y diseño.

Aplicación del lenguaje de Modelado Unificado para facilitar el análisis y diseño de programas orientados a objetos.

El patrón de arquitectura MVC que permite separar la aplicación en una estructura eficaz pero simple.

Una buena aplicación debe ser cuidadosamente planeada y desarrollada antes de escribir siquiera una sola línea de código. El análisis y diseño orientado a objeto son las tareas más duras que tendremos que afrontar, son muchos a los que no les resulta fácil adquirir destreza en este campo. No obstante, es importante que nos esforcemos al máximo para dominarlos completamente.

Nuestro camino sólo ha empezado, la siguiente fase consiste en ganar experiencia.

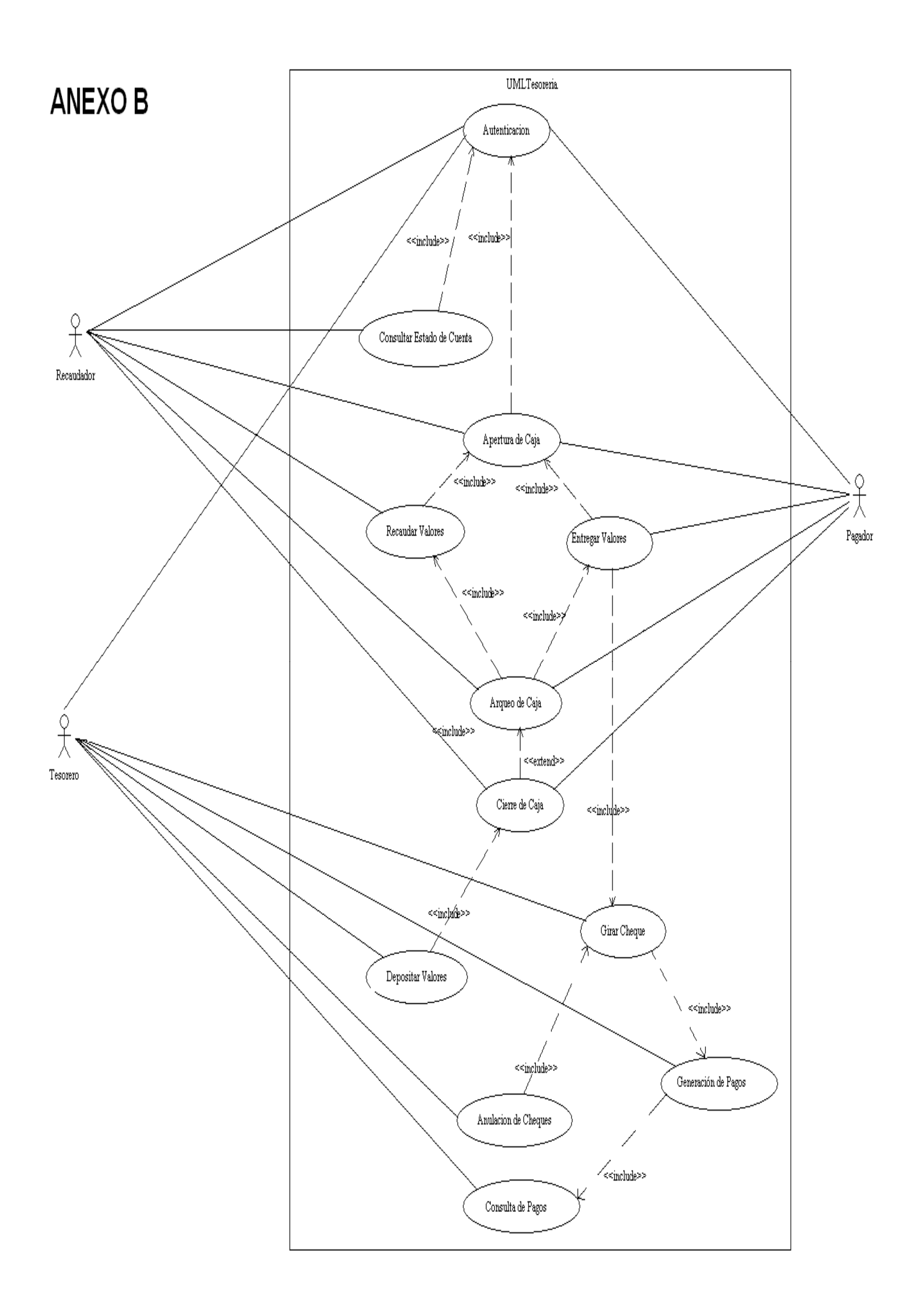
ANEXOS

ANEXO A

Actividades: Levantamiento de información; Auto capacitación; Asesoramiento profesional; Análisis; Documentación del análisis; Diseño; Documentación del diseño; Implementación; Documentación de la implementación; Pruebas y depuración; Documentacion pruebas, manual técnico y de usuario; Sustentaciones previas y final

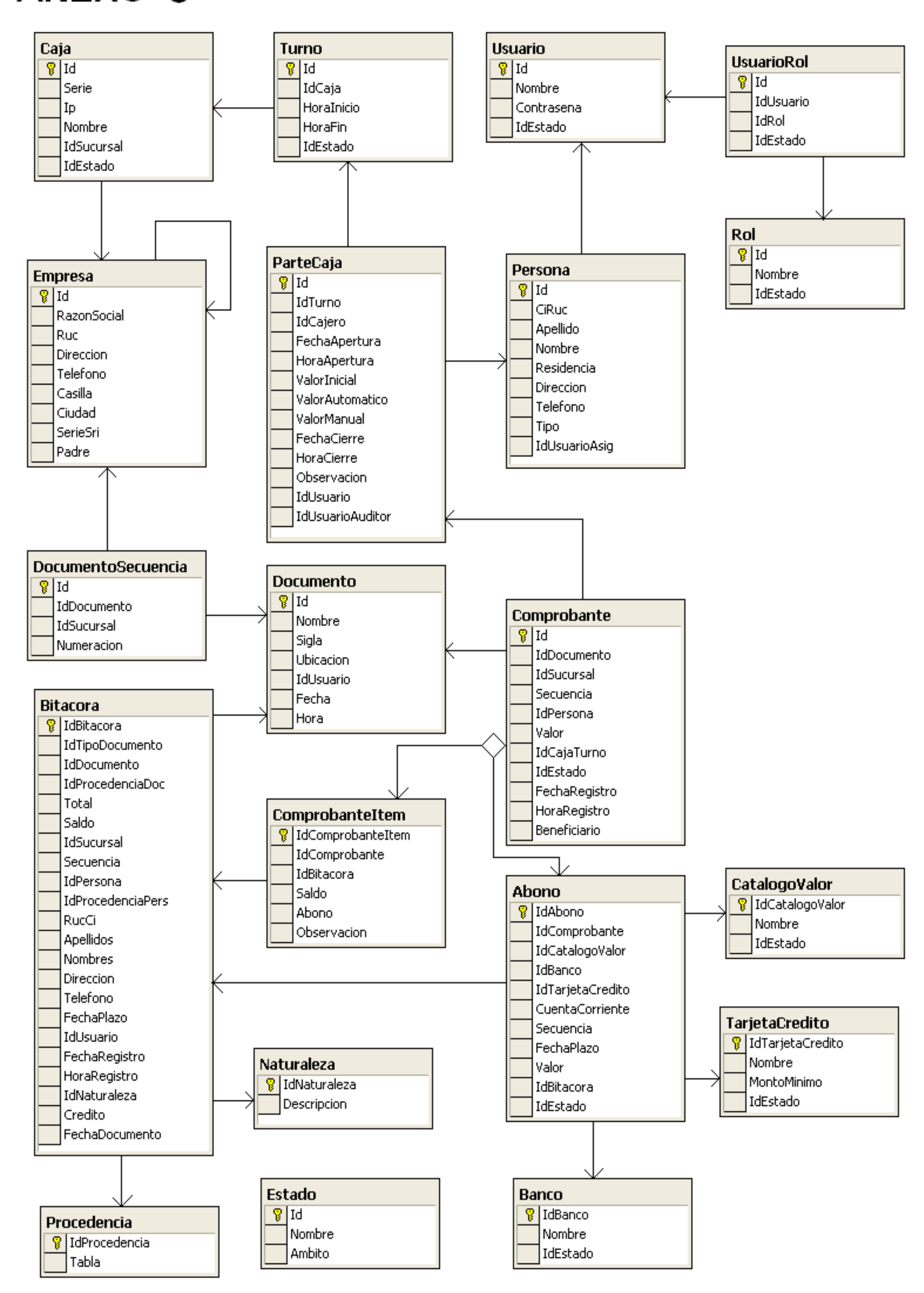
Mes	Día	Fecha
ENERO	L	23
	M	24
	M	25
	J	26
	V	27
	S	28
	D	29
	L	30
	M	31
FEBRERO	M	1
	J	2
	V	3
	S	4
	D	5
	L	6
	M	7
	M	8
	J	9
	V	10
	S	11
	D	12
	L	13
	M	14
	M	15
	J	16
	V	17
	S	18
	D	19
	L	20
	M	21
	M	22
	J	23
	V	24
	S	25
	D	26
	L	27
	M	28
MARZO	M	1
	J	2
	V	3
	S	4
	D	5
	L	6
	M	7
	M	8
	J	9
	V	10
	S	11
	D	12
	L	13
	M	14
	M	15
	J	16
	V	17
	S	18
	D	19
	L	20
	M	21
	M	22
	J	23
	V	24
	S	25
	D	26
	L	27
	M	28
	M	29
	J	30
	V	31
ABRIL	S	1
	D	2
	L	3
	M	4
	M	5
	J	6
	V	7
	S	8
	D	9
	L	10
	M	11
	M	12
	J	13
	V	14
	S	15
	D	16
	L	17
	M	18
	M	19
	J	20
	V	21
	S	22
	D	23
	L	24
	M	25
	M	26
	J	27
	V	28
	S	29
	D	30
MAYO	L	1
	M	2
	M	3
	J	4
	V	5
	S	6
	D	7
	L	8
	M	9
	M	10
	J	11
	V	12
	S	13
	D	14
	L	15
	M	16
	M	17
	J	18
	V	19
	S	20
	D	21
	L	22
	M	23
	M	24
	J	25
	V	26
	S	27
	D	28
	L	29
	M	30
	M	31
JUNIO	J	1
	V	2
	S	3
	D	4
	L	5
	M	6
	M	7
	J	8
	V	9
	S	10
	D	11
	L	12
	M	13
	M	14
	J	15
	V	16
	S	17
	D	18
	L	19
	M	20
	M	21
	J	22
	V	23
	S	24
	D	25
	L	26
	M	27
	M	28
	J	29
	V	30
JULIO	S	1
	D	2
	L	3
	M	4
	M	5
	J	6
	V	7
	S	8
	D	9
	L	10
	M	11
	M	12
	J	13
	V	14
	S	15
	D	16
	L	17
	M	18
	M	19
	J	20
	V	21
	S	22
	D	23
	L	24
	M	25
	M	26
	J	27
	V	28
	S	29
	D	30
	L	31
AGOSTO	M	1
	M	2
	J	3
	V	4
	S	5
	D	6
	L	7
	M	8
	M	9
	J	10
	V	11
	S	12
	D	13
	L	14
	M	15
	M	16
	J	17
	V	18
	S	19
	D	20
	L	21
	M	22
	M	23
	J	24
	V	25
	S	26
	D	27
	L	28
	M	29
	M	30
	J	31
SEPTIEMBRE	V	1
	S	2
	D	3
	L	4
	M	5
	M	6
	J	7
	V	8
	S	9
	D	10
	L	11
	M	12
	M	13
	J	14
	V	15
	S	16
	D	17
	L	18
	M	19
	M	20
	J	21
	V	22
	S	23
	D	24
	L	25
	M	26
	M	27
	J	28
	V	29
	S	30
OCTUBRE	D	1
	L	2
	M	3
	M	4
	J	5
	V	6
	S	7
	D	8
	L	9
	M	10
	M	11
	J	12

ANEXO B

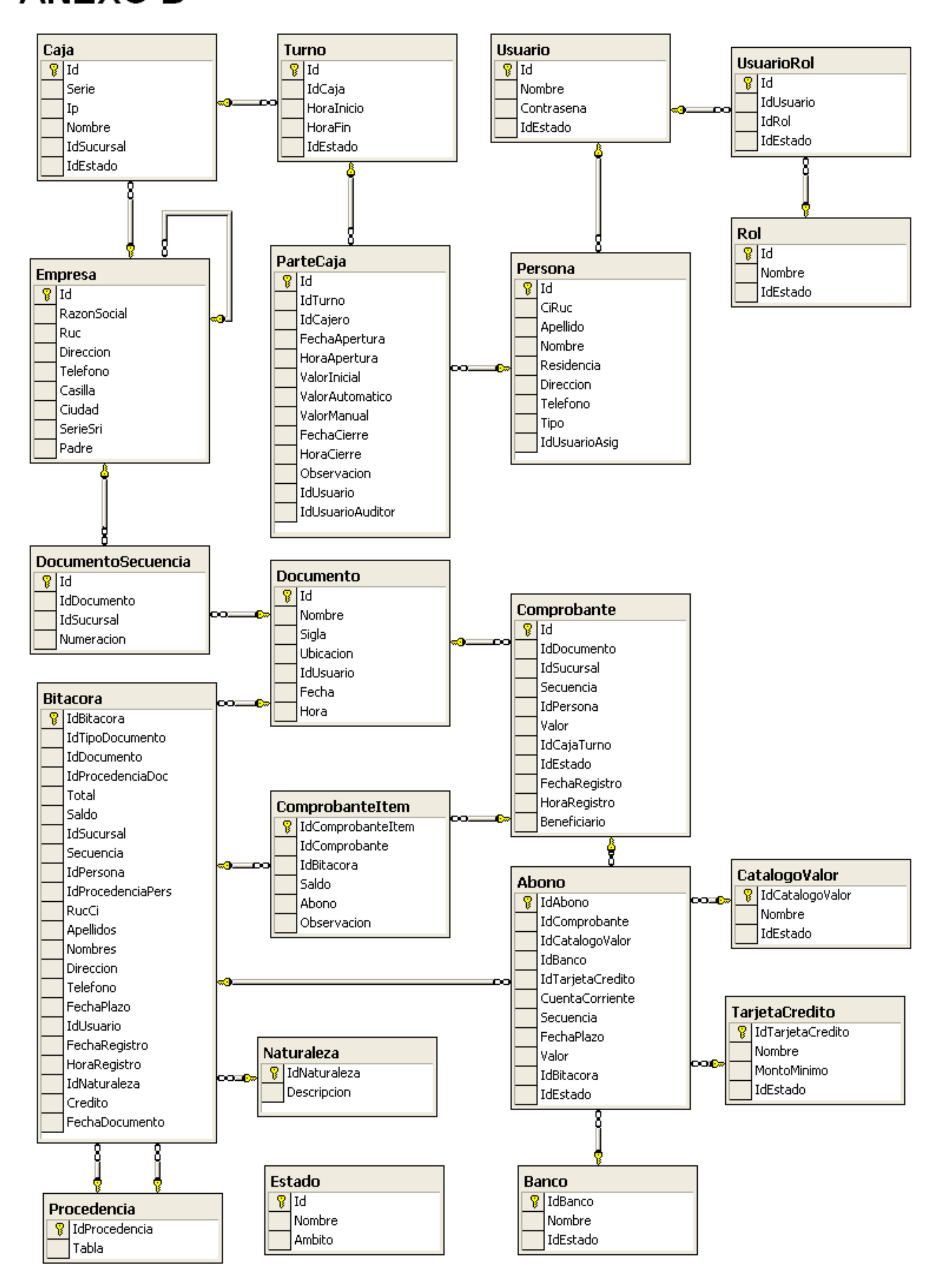


UMLTesoreria
Autenticacion
<<include>>
<<include>>
Consultar Estado de Cuenta
Recaudador
Apertura de Caja
<<include>>
<<include>>
Recaudar Valores
Entregar Valores
Pagador
<<include>>
<<include>>
Arqueo de Caja
<<include>>
<<extend>>
Tesorero
Cierre de Caja
<<include>>
<<include>>
Girar Cheque
Depositar Valores
<<include>>
<<include>>
Generación de Pagos
Anulacion de Cheques
<<include>>
Consulta de Pagos

ANEXO C


Caja
Id
Serie
Ip
Nombre
IdSucursal
IdEstado
Turno
Id
IdCaja
HoraInicio
HoraFin
IdEstado
Usuario
Id
Nombre
Contrasena
IdEstado
UsuarioRol
Id
IdUsuario
IdRol
IdEstado
Rol
Id
Nombre
IdEstado
Empresa
Id
RazonSocial
Ruc
Direccion
Telefono
Casilla
Ciudad
SerieSri
Padre
ParteCaja
Id
IdTurno
IdCajero
FechaApertura
HoraApertura
ValorInicial
ValorAutomatico
ValorManual
FechaCierre
HoraCierre
Observacion
IdUsuario
IdUsuarioAuditor
Persona
Id
CiRuc
Apellido
Nombre
Residencia
Direccion
Telefono
Tipo
IdUsuarioAsig
DocumentoSecuencia
Id
IdDocumento
IdSucursal
Numeracion
Documento
Id
Nombre
Sigla
Ubicacion
IdUsuario
Fecha
Hora
Comprobante
Id
IdDocumento
IdSucursal
Secuencia
IdPersona
Valor
IdCajaTurno
IdEstado
FechaRegistro
HoraRegistro
Beneficiario
Bitacora
IdBitacora
IdTipoDocumento
IdDocumento
IdProcedenciaDoc
Total
Saldo
IdSucursal
Secuencia
IdPersona
IdProcedenciaPers
RucCi
Apellidos
Nombres
Direccion
Telefono
FechaPlazo
IdUsuario
FechaRegistro
HoraRegistro
IdNaturaleza
Credito
FechaDocumento
ComprobanteItem
IdComprobanteItem
IdComprobante
IdBitacora
Saldo
Abono
Observacion
Abono
IdAbono
IdComprobante
IdCatalogoValor
IdBanco
IdTarjetaCredito
CuentaCorriente
Secuencia
FechaPlazo
Valor
IdBitacora
IdEstado
CatalogoValor
IdCatalogoValor
Nombre
IdEstado
TarjetaCredito
IdTarjetaCredito
Nombre
MontoMinimo
IdEstado
Naturaleza
IdNaturaleza
Descripcion
Procedencia
IdProcedencia
Tabla
Estado
Id
Nombre
Ambito
Banco
IdBanco
Nombre
IdEstado

ANEXO D


Caja
Id
Serie
Ip
Nombre
IdSucursal
IdEstado
Turno
Id
IdCaja
HoraInicio
HoraFin
IdEstado
Usuario
Id
Nombre
Contrasena
IdEstado
UsuarioRol
Id
IdUsuario
IdRol
IdEstado
Rol
Id
Nombre
IdEstado
Empresa
Id
RazonSocial
Ruc
Direccion
Telefono
Casilla
Ciudad
SerieSri
Padre
ParteCaja
Id
IdTurno
IdCajero
FechaApertura
HoraApertura
ValorInicial
ValorAutomatico
ValorManual
FechaCierre
HoraCierre
Observacion
IdUsuario
IdUsuarioAuditor
Persona
Id
CiRuc
Apellido
Nombre
Residencia
Direccion
Telefono
Tipo
IdUsuarioAsig
DocumentoSecuencia
Id
IdDocumento
IdSucursal
Numeracion
Documento
Id
Nombre
Sigla
Ubicacion
IdUsuario
Fecha
Hora
Comprobante
Id
IdDocumento
IdSucursal
Secuencia
IdPersona
Valor
IdCajaTurno
IdEstado
FechaRegistro
HoraRegistro
Beneficiario
Bitacora
IdBitacora
IdTipoDocumento
IdDocumento
IdProcedenciaDoc
Total
Saldo
IdSucursal
Secuencia
IdPersona
IdProcedenciaPers
RucCi
Apellidos
Nombres
Direccion
Telefono
FechaPlazo
IdUsuario
FechaRegistro
HoraRegistro
IdNaturaleza
Credito
FechaDocumento
ComprobanteItem
IdComprobanteItem
IdComprobante
IdBitacora
Saldo
Abono
Observacion
Abono
IdAbono
IdComprobante
IdCatalogoValor
IdBanco
IdTarjetaCredito
CuentaCorriente
Secuencia
FechaPlazo
Valor
IdBitacora
IdEstado
CatalogoValor
IdCatalogoValor
Nombre
IdEstado
TarjetaCredito
IdTarjetaCredito
Nombre
MontoMinimo
IdEstado
Naturaleza
IdNaturaleza
Descripcion
Procedencia
IdProcedencia
Tabla
Estado
Id
Nombre
Ambito
Banco
IdBanco
Nombre
IdEstado

BIBLIOGRAFÍA

- Clark D.; Introducción a la Programación Orientada a Objetos con Visual Basic.Net; Ediciones Anaya Multimedia; Madrid – España, 2003; Capítulos 2, 3, 4.

- Falkner J., Galbraith B., Irani R., Kochmer C., Narayana S., Perrumal K. Jimney J. ; Fundamentos Desarrollo Web con JSP ; Ediciones Anaya Multimedia; Madrid – España, 2002.

- Contraloría General del Estado; Normas de Control Interno para el Sector Público de la República del Ecuador; Quito – Ecuador, 2002; Páginas 37 – 42.

- Alvarez N.; Curso Básico de Contabilidad; Editorial Mc Graw Hill; Segunda Edición.

- www.sri.gov.ec